Intent Role Oriented Query Parsing and Its Application in Subtopic Mining

Yu Haitao

A thesis submitted to The University of Tokushima in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy

September, 2013

The University of Tokushima
Graduate School of Engineering
Information Science and Systems Engineering

Abstract

Nowadays, web search engines are playing an increasingly dominant role in people's daily information access on the web. But the majority of users are submitting short queries, many of which are ambiguous and/or underspecified. Thus, understanding the underlying intent or information need encoded within a query has become an essential step for effective information retrieval.

Based on the statement that the composing terms play different roles in manifesting the information need or intent encoded in a query, this thesis proposes the strategy of intent role oriented query parsing. Two intent roles, *kernel-object* and *modifier*, are defined, based on which queries are further classified as *role-explicit queries* and *role-implicit queries*.

In the so-called intent role oriented query parsing, several fundamental issues are studied:

(i) *Role-explicit query extraction*, which aims to extract a repository of role-explicit queries. In particular, this thesis separates all sessions in a query log into mul-sessions and sin-sessions. A number of features in the context of a session are used to quantify the weight of a term being the kernel-object. Furthermore, according to the session type, different approaches are proposed to extract role-explicit queries.

(ii) *Intent role annotation*, which refers to identifying the composing kernel-object and modifier of a given role-explicit query. This thesis proposes the simplified n-gram role language model to perform intent role annotation, which achieves a satisfactory performance, more than 73% in terms of the term-level and query-level metrics.

(iii) *Role-explicit query identification*, which refers to determining whether or not an arbitrary query is a role-explicit query. Using a set of predictive features, this thesis investigates a number of machine learning classifiers to perform role-explicit query identification. The experimental results show that the classifiers can achieve more than 90% precision in identifying role-explicit queries.

Finally, this thesis investigates how intent role oriented query parsing can be adapted and integrated into the subtopic mining task. Intuitively, the concept of modifier graph is proposed. When the edges are weighted using the term-level knowledge in query log,

the clustering of a modifier graph uncovers the underlying subtopics with respect to a particular topic. Then the problem of subtopic mining is reduced to the clustering of modifier graph. Based on this idea, the strategy of subtopic mining via modifier graph clustering is proposed. Leveraging on the standard test collection and data released by NTCIR-10, a series of experiments are conducted. The experimental results show that the proposed approaches based on modifier graph clustering are robust to the sparseness problem and outperforms the two baseline methods in terms of *I-rec*, *D-nDCG* and *D#-nDCG*.

The central contributions of this thesis are the introduction of intent role oriented query parsing and its application in subtopic mining. In deed, the intent role oriented query parsing is worthy to be further explored in many other applications, e.g., geographic intent analysis, document re-ranking in information retrieval.

Acknowledgements

I wish to express my sincere thanks to all the people who game me tremendous support and help during my Ph.D study.

First and foremost, I am thankful to my advisor, Professor Ren Fuji. Without his guidance and support, this thesis would not have been possible. Besides providing the pleasant research environment, he has taught me lots of skills in multiple aspects, from which I would benefit for my whole life. His hard work spirit and dedication encourage me to cross the barriers and move forward in the future.

I also thank the other thesis committee members, Professor Kenji Terada and Professor Masami Shishibori. They have spent so much precious time in reviewing this thesis. Their constructive comments are invaluable for both the thesis and my career.

I am also grateful to the staffs in the Education Office of the Embassy of PRC (People's Republic of China) in OSAKA and TOKYO for their kind helps. I would like to give my great thanks to the China Scholarship Council (CSC) that gave me fund support for studying abroad. With this support, I am able to focus on my research and complete this thesis.

It is really my honor to be a member of A1-Lab. The discussions here gave me inspiration for my research. Thanks to Kang Xin, Liu Song, Wang Jun, Zhang Guodong, Wu Ye, Li Ji, Quan Changqin, Sun Yan, Wang Cheng, Mu Rong, Zhang Wei, Wang Lei, Sun Xiao, Sohrab, Cao Mengsi, Li Bo, Feng Xiaobo, Nie Yang, Liu Wenjing, etc. All the professors and staffs in the Department of Information Science & Intelligent Systems and International Center are very kind and patient, such as Professor Jin Chenghai, Section-chief Fujikawa-san and Staff Sato-san, all of them make up a comfortable campus-life, especially for foreign students.

I am proud to work together with the members in Association of Chinese Students and Scholars in Tokushima, such as Mu Rong, Xiao Qingmei, Zhang Hao, Xu Xiaojing, Li Rongrong, Chen Mei, Sun Yuanzhi, Zhang Bo, Sun Yiyin, Liu Huan, Wang Han, Hou Yanfei, etc. It really was a good experience and I did learn a lot from you all. Besides, the members in Japan-China Friendship Association of Tokushima, such as Ikuta-sensei

and Kakihara-sensei, provide us opportunities to organize activities together, which helps us to get more familiar with Japanese culture.

Last but not least, I would like to given my heartful thanks my parents, my sister and other relatives for their love and confidence in me. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The advent of the Word Wide Web (WWW, commonly known as the web) puts forth the inception of the information era. The overwhelming scale of web pages, reports, emails and spreadsheets do facilitate modern knowledge transfer. Likewise, it has also caused an information explosion, which makes it hard for common web users to find the desired information quickly. As an efficient search tool, the web search engine is designed to assist users to search information on the web. Figure 1.1 roughly depicts the procedures of information search via a web search engine.

Figure 1.1: Information search via a web search engine.

As shown in Figure 1.1, in order to meet the online requirements like providing relevant documents in response to a particular request in real time, the web search engine performs several processes ahead of time (typically known as *crawling* and *indexing*). For web users, they manifest their information needs in the form of a sequence of terms, which

is commonly called a *query*, and then browse the returned result list for desired information. If they are not satisfied with the results in response to the previous query, a reformulated query (known as *reformulation* or *refinement*) will be conceived and submitted again. For example, a user may enter a query *Harry Potter*, but subsequently submits a query refinement *Harry Potter game* when he/she is unsatisfied with the results with respect to *Harry Potter*. Moreover, with the evolution of web search engines, sponsored advertisements (ads) are also provided on the marginal regions of a search engine result page. Users with commercial intents are likely to click the sponsored ads and be navigated to specific websites [27, 29, 49].

Modern web search engines are more complicated, which commonly consist of tens of independent modules. For a good understanding of the techniques used in a web search engine system, the materials [7, 60] are good starting point references.

Recent studies show that billions of daily searches are made by web users [28]. Instead of formulating natural language queries, the vast majority of users are submitting short queries, which generally show no grammatical or syntactical structure. It would be error prone if we directly project standard natural language processing strategies into query analysis. And what's more the short queries are often ambiguous and/or underspecified. For example, for the query *Harry Potter*, it could refer to a book, a movie or a game software. In the category of movie, the user may be interested in the main character or the reviews. This problem is more crucial for east asian languages, e.g., Chinese, Japanese and Korean, which is written with no term delimiters. Therefore, it is hard for web search engines to accurately capture the information need encoded within a search query and provide desired results at a stroke.

In fact, understanding the information need or intent encoded within a search query has long been regarded as an essential step for effective information retrieval. Towards this goal, great efforts have been made by academic researchers, communities, as well as the web search engine companies. Different techniques have been proposed and investigated, e.g., query segmentation [44, 59, 63, 78, 93, 107], query recommendation [12, 14, 18, 90, 91, 101, 108], query annotation [5, 6, 58, 61], etc. Reasons resulting in this problematic issue can be classified as: (1) Search engines are often used to search unfamiliar topics and users may be unaware of the exact terms matching their information needs; (2) Queries

are often submitted in the absence of any knowledge of user's preference or context; (3) The senses of submitted terms may be used in ways beyond the original imagination of users.

By learning lessons from previous studies, we look into user queries at a fine-grained term level. Based on the observation that the utilities of terms within a query are different when being selected to manifest a specific information need, we introduce two intent roles (*kernel-object* and *modifier*) to differentiate terms' different utilities. Leveraging on the proposed intent roles, we further classify user queries into two types: role-explicit query and role-implicit query, which enables us to devise different approaches for query understanding according to the query type. The proposed scheme of intent role annotation not only helps to understand users' intent representation, but also paves the way for capturing the knowledge (also known as the wisdom of crowds) hidden in a large-scale query log.

1.2 Contributions

The main contributions of this thesis are the following:

For better query representation and understanding, we propose two intent roles (*kernel-object* and *modifier*) to structurally parse user queries. Though simple, it helps to interpret the way of how a user generally formulates a query. For instance, under intent role oriented query parsing, users' query formulating behavior can be simplified as: The user firstly conceives the kernel-object that abstracts the core object or topic of his/her information need, some (kernel-object)-dependent modifiers are selected subsequently, then a query is formulated. To our knowledge, we are the first that analyzes user queries through intent roles as defined in this thesis.

Next, we study how to extract role-explicit queries from a given query log. An unsupervised system that can automatically extract a repository of sufficient role-explicit from a query log is proposed. A series of experiments are conducted based on the query log SogouQ, which demonstrates the effectiveness of the proposed system for role-explicit query extraction.

Later in the thesis, we study another two fundamental issues when performing intent

role oriented query parsing, namely intent role annotation and role-explicit query identification. For intent role annotation, the simplified n-gram role language model is proposed. As for role-explicit query identification, we propose to address it using machine learning classifiers. A set of predictive features are proposed and tested.

Finally, we investigate how intent role oriented query parsing can be adapted to facilitate query understanding. In particular, we propose the strategy of subtopic mining via modifier-graph clustering, the core of which is intent role oriented query parsing. Based on the standard test collections released by NTCIR-10 workshop, a series of experiments are conducted. The experimental results demonstrate that the strategy of modifier-graph based subtopic mining is effective and robust to the sparseness problem that many methods suffer from.

1.3 Thesis Outline

This thesis consists of 7 chapters, which are organized as follows:

In Chapter 1, the background of query representation and understanding is depicted.

In Chapter 2, the main related techniques are introduced, such as query segmentation, named entity recognition in query, query annotation and query intent inference. In particular, the typical algorithms, as well as the evaluation metrics, are described. Finally, the pros and cons of the prior studies are discussed.

In Chapter 3, the intuition behind the intent roles is detailed. Leveraging on the intent roles, we interpret how a user generally formulate the query to manifest his/her information need. Moreover, a preliminary discussion of intent role oriented query parsing is presented.

In Chapter 4, we study how to extract role-explicit queries from a given query log. In particular, the overlapping feature (e.g., the longest common substring) in the context of mul-session is explored to quantify the certainty of a term being the kernel-object. An unsupervised system that performs role-explicit query extraction concurrently with intent role annotation is proposed.

In Chapter 5, we study two fundamental problems (intent role annotation and role-explicit query identification) when performing intent role oriented query parsing. The

simplified n-gram role language model is proposed for intent role annotation. We solve the problem of role-explicit query identification in a supervised manner. Different machine learning methods are compared. Finally, a series of experiments are conducted to test the effectiveness of the proposed model for intent role annotation and classifiers for role-explicit query identification.

In Chapter 6, we propose the strategy of subtopic mining via modifier graph clustering, which is an extended usage of intent role oriented query parsing. In particular, we show how subtopic mining can be reduced to that of graph clustering over the modifier graph. A series of experiments are conducted to test the strategy of modifier-graph based subtopic mining based on the standard test collection released by NTCIR-10 workshop.

In Chapter 7, we close this thesis with the conclusions drawn from this work, as well as interesting future research directions.

Chapter 2

Related Work

2.1 Introduction

In this chapter, we survey the techniques proposed for query parsing and understanding. Due to its essential importance to effective information retrieval, query parsing and understanding indeed is a very broad research topic and has been widely investigated from diverse dimensions. Because of no standard way of describing these techniques, we survey the typical techniques in terms of *query segmentation*, *named entity recognition in query*, *query annotation*, *query intent inference*. Of course, the techniques proposed for query parsing and understanding are not limited to these fields. However, one thing in common is that all of these techniques are query-centric.

2.2 Query Segmentation

2.2.1 Problem Formulation

The task of query segmentation is to divide a query into disjoint terms, each term corresponds to a semantic unit. Formally, for an English query consisting of a sequence of words, the query segmentation can be given as:

$$q = w_1 w_2 ... w_n \rightarrow q = t_1 ... t_k \tag{2.1}$$

where w denotes a word, t denotes a term.

For an east asian language query (e.g., Chinese) that consists of a sequence of characters, the query segmentation can be given as:

$$q = ch_1 ch_2 ... ch_m \rightarrow q = t_1 ... t_j \tag{2.2}$$

where ch denotes a character.

A variety of algorithms for query segmentation have been proposed. Based on whether an algorithm requires the training data or not, we classify the query segmentation algorithms into unsupervised algorithms and supervised algorithms.

2.2.2 Unsupervised Query Segmentation

Unsupervised query segmentation [44, 59, 63, 78, 93, 107] requires no training data. Some algorithms merely rely on a set of queries itself, some algorithms incorporate the knowledge derived from external resources, such as Wikipedia and Google n-gram corpus. In the following, we introduce several typical unsupervised segmentation algorithms.

Under that assumption that the composing terms of a query are independent and identically-distributed (I.I.D.), Tan et al. [93] proposed a generative model to recover a query's underlying terms. By optimizing the minimum description length of a partial corpus with respect to a particular query, the model's parameters are estimated with the expectation-maximization (EM) algorithm. To further augment the unsupervised segmentation, they used the *article titles* in Wekipedia as the evidence that a segment of a query is a well-formed unit.

Hagen et al. [44] proposed the algorithm of naive query segmentation. Leveraging on the Google n-gram corpus, this algorithm derives a score for each valid segmentation of a query. Finally the segmentation that achieves the maximum score is selected as the optimal segmentation. Moreover, they suggested to use the *article titles* in Wekipedia to normalize a segment's frequency.

Mishra et al. [63] proposed an algorithm that merely relies on a set of queries, in which the Hoeffding's Inequality is incorporated. Similar to Hagen et al. [44], for each possible segmentation of the query, a segmentation score is computed and the one that yields the maximum segmentation score is selected as the optimal segmentation. Furthermore, Saha

et al. [78] studied how to augment this algorithm with Wikipedia titles.

Instead of using a large text corpus, Li et al. [59] proposed a probabilistic model for query segmentation leveraging on the clickthrough data. This model derives from the observation that the clicked documents with respect to a particular query reflect the original user's preferences of how to divide the query. Their experimental results have shown that this segmentation model not only outperforms the baseline methods, but also helps to improve the retrieval relevance.

2.2.3 Supervised Query Segmentation

As stated in Section 2.2.1, the training data is the premise of supervised query segmentation. For example, Yu and Shi [105] focused on queries stored in a relational database and studied how to segment queries using the conditional random fields (CRF) model.

Supervised query segmentation suffers from the following shortcomings: (1) A small size of training data would result in a poor performance, while a large scale of training data would be labor intensive and unaffordable; (2) Due to the limited knowledge learned from the training data, the supervised query segmentation algorithm can't work well with respect to queries from different domains. This is also the reason why researchers mostly focus on unsupervised query segmentation algorithms.

2.2.4 Evaluation Metric

A quality query segmentation algorithm features the following two aspects: (1) *Efficiency.* The runtime is crucial because query segmentation must be done on-the-fly, especially when being integrated into the real utilization like information retrieval. (2) *Accuracy.* Namely the output of terms should be as accurate as possible.

To measure the segmentation accuracy, there are metrics at different levels [44, 93]:

- *Break Level:* As query segmentation can also be treated as a classification problem, i.e., whether or not to add a segment break between each pair of consecutive words (e.g., English queries) or characters (e.g., Chinese queries). In particular, there are $n-1$ potential break positions for a query with n characters or words. For each break position, the output segmentation may either make a correct decision or a

false one compared with the ideal segmentation. The *break level accuracy* is defined as the ratio of correct decisions.

- *Term Level:* Let ∇ be the set of terms of the ideal segmentation of a query, $\nabla^{'}$ be the terms of the computed segmentation. If we treat ∇ as a set of "relevant" terms, we aim to measure how well the computed segmentation recovers these terms. Thus the widely used metrics (precision, recall and f-score) in the field of information retrieval can be analogously given as:

$$T_{pre} = \frac{\nabla \cap \nabla^{'}}{\nabla^{'}}, \quad T_{rec} = \frac{\nabla \cap \nabla^{'}}{\nabla}, \quad T_{f-score} = \frac{2 * T_{pre} * T_{rec}}{T_{pre} + T_{rec}} \tag{2.3}$$

- *Query Level:* The query level metric becomes more rigid in judging how well the predicted segmentation matches the ideal segmentation at the whole query granularity. A segmentation is viewed as correct iff it exactly consists of the same terms as the ideal segmentation. Thus, the *query level accuracy* is defined as the ratio of correctly segmented queries again a test collection. Let Δ be the set of ideal segmentations of the test collection, $\Delta^{'}$ be the computed segmentations, the query level accuracy is given as:

$$Q_{pre} = \frac{\Delta \cap \Delta^{'}}{\Delta^{'}} \tag{2.4}$$

2.3 Named Entity Recognition in Query

2.3.1 Problem Formulation

As shown in the study by Guo et al. [43], about 70% of search queries contain named entities. If the named entities within a query can be correctly identified, it does help to understand search intents better and thus provide better search.

In view of the above-mentioned benefit, the task of named entity recognition in query (NERQ) is formulated as: Given a query, we wish to detect the named entities within the query and categorize the named entities into classes from a predefined taxonomy. Moreover, the study by Guo et al. [43] showed that if a query contains named entities, it

is common that single named entity occurs and less than 1% of the queries contain two or more named entities. Therefore, most of the researchers only take single-named-entity queries into account.

2.3.2 Algorithms for Named Entity Recognition in Query

In the study by Guo et al. [43], the remaining parts of a query by removing the named entity is viewed as the context of a named entity, the class of the named entity is interpreted as topics. Thus the topic model (Latent Dirichlet Allocation) can be constructed. To ensure the alignment between model topics and predefined classes, a weakly supervised learning method was further proposed. In a similar manner, Xu et al. [100] also explored the weakly supervised Latent Dirichlet Allocation method for named entity recognition in query.

Based on the observation that: in most of the cases, the user queries within the same search session are topically relevant, Du et al. [34] incorporated the *class feature* and *overlap feature* (extracted from search sessions) into the Conditional Random Field (CRF) model for identifying the named entities.

In a different manner, Paşca [68] studied how to extract named entities using template based method. Given a small set of seed named entities, named entities pertaining to diverse classes of interest can be obtained.

2.4 Query Annotation

2.4.1 Problem Formulation

By query annotation, we refer to the task that parses a query through annotating its composing units with category labels or a set of pre-defined labels.

Compared query segmentation (Section 2.2), NERQ (Section 2.3) not only identifies the main semantic unit, but also assigns a proper class attribute. In contrast, most query annotation techniques take each composing unit of a query into account, and assign a pre-defined label or class attribute.

2.4.2 Algorithms for Query Annotation

Some typical query annotation algorithms are introduced as follows:

Barr et al. [6] investigated the applicability of part-of-speech (POS) tagging to English-language queries, and its potential value of POS tagging for improving search performance. Similarly, Bendersky and Croft [9] focused the shallow linguistic structure of a query. Three annotation sequences (Capitalization, POS tags and Segmentation indicator) are used to parse a query.

To capture the aspects of user's information need, Bailey et al. [5] used the category labels of Open Directory Project (ODP) to annotate long and rare queries. And their experimental results showed that the annotated labels are useful for improving the retrieval performance, e.g., by re-ordering the top-ranked results.

By assuming that queries are already classified into correct domains, Manshadi and Li [61] used domain-specific labels to annotate each term within a query. Take the instance shown in their work as an example, the query *cheap garmin streetpilot c340 gps* in the *product* domain will be annotated as:

cheap	garmin	streetpilot	c340	gps
|	|	|	|	|
SortOrder	Brand	Model	Model	Type

As the queries studied by Manshadi and Li [61] mainly are *noun phrase queries*, i.e., the queries that contain one or more noun phrases, Li [58] further used the concepts of *intent head* and *intent modifier* to parse queries of this kind. Supervised and semi-supervised models were proposed to perform the target annotation.

2.5 Query Intent Inference

2.5.1 Problem Formulation

As discussed in Section 1.1, though entering the same query *Harry Potter*, different users may prefer different information. For example, a book, a movie or a game software. In the category of movie, some users may be interested in the main character, some may look for the reviews. In the field of query understanding, a number of concepts are used to

denote one information need underlying a query, e.g., *intent* [73, 77], *subtopic* [46, 79, 89]. In this thesis, these two concepts are used equivalently.

By query intent inference, we refer to the task of identifying the possible intents or subtopics for a given query. The organization of the identified intents or subtopics may differ due to different methods. The task of subtopic mining in Chapter 6 can be viewed as another formulation of query intent inference.

2.5.2 Query Log Mining

"History teaches everything, even the future."

—Alphonse de Lamartine, speech at Macon 1847.

The statement written by Alphonse de Lamartine in the ninetheen century points out the value of historical information. Analogously, query log records the historical search behaviors performed by massive users, it contains invaluable latent knowledge that can be successfully used in many aspects, such as:

- **Query Suggestion (or Query Recommendation)**. [12, 14, 18, 90, 91, 101, 108] derive knowledge from query log for generating a list of alternative queries to help a user express his or her information need. The suggested queries are semantically related or are frequently entered by a large number of users. A user can immediately click the one that approximately represents his or her information need. As a simple form of interactive search, query suggestion has been widely utilized. Examples in real life include the service provided by Google, Bing, Yahoo! Search, etc.

- **Query Intent Inference**. To capture the underlying intent or information need of a query, [4, 46, 73, 77, 103] derive knowledge from query log for determining the semantic relations among queries. In general, queries expressing a similar or the same information need are grouped into one cluster, which denotes one particular intent.

- **Document Ranking**. As a long result list is time-consuming and hard to browse, users expect to find their desired information in the first result page as far as possible.

For higher degree of satisfaction, [1, 19, 20, 51, 55, 74, 94] derive knowledge from query log for generating a quality result list.

Query log mining is a hot research topic nowadays. Besides the above aspects, lots of studies have been conducted from many other dimensions by academic researchers, as well as major search engine companies. The proceedings of the top conferences (e.g., SIGIR, WWW, CIKM, WSDM, KDD, VLDB, ACL, EMNLP) and the journals (e.g., ACM TWEB, ACM TOIS, ACM TKDD) cover the major and up-to-date techniques in the field of query log mining. For a general overview, [87] is a good starting point. In this thesis, we mainly focus on the methods for query intent inference.

2.5.3 Algorithms for Query Intent Inference

To capture the underlying information needs encoded within user queries, considerable studies have been conducted from various aspects.

Beeferman and Berger [8] viewed the query log as a bipartite graph, and the obtained clusters were interpreted as topics covering diverse queries. The research by Jones and Klinkner [52] showed that users' search tasks are interleaved or hierarchically organized. They studied how to segment sequences of user queries into a hierarchical structure. Sadikov et al. [77] and Radlinski et al. [73] tried to infer the underlying information needs of a query by clustering its refinements. The web page click and session co-occurrence information are viewed as indicators of relevance. Chien et al. [23] measured the query similarity through temporal correlations, which reduces the need to understand queries at a linguistic level. A key feature adopted by Hu et al. [46] was that: many users add one or more additional keywords to expand the query to clarify their search intents. In their study, a subtopic was represented by keywords and URLs.

Yin and shah [103] focused on how to enhance user experiences for named entity queries and proposed algorithms to build a hierarchical taxonomy of generic intents at a class level, where the words or phrases co-appearing with entities are used to represent generic intents. E.g., *albums*, *songs* and *music videos* for the *musician* class, which are generic to *Britney Spears*, *Michael Jackson*, etc. Similarly, Wang et al. [96] extracted broad query aspects that are applicable to a class of queries to help users accomplish

query reformulation. However, there are cases where we need to go beyond the generic intents or broad aspects at a hard class level. E.g., both *Mozart* and *Avril* are famous persons, a significant portion of users commonly anticipate their specific works rather than the generic intents.

In particular, [97, 102] studied problem of geographic intent (Geo intent) understanding, especially a class of queries that include explicit location terms. For example, *Manhattan coffee* is regarded as a query with explicit Geo intent. They parsed a query with an Geo intent into two parts: a location part and a non-location part. They learned models from non-location parts of queries with explicit Geo intents to detect users' Geo information needs.

2.6 Summary

Based on the above survey, we can see that:

(1) Query segmentation is a preliminary but an essential step for query parsing and understanding. Many algorithms in the fields of query annotation and query intent inference rely on the output of query segmentation.

(2) Many algorithms (e.g., [58, 61, 96, 103]) in the fields of named entity recognition and query annotation restrict the recognition or annotation to a set of pre-defined or pre-collected classes. When parsing overwhelming web queries in real life, they would suffer from the requirement of a fine-grained classification, e.g., named entity classification. In fact, defining fine-grained types for entities on the web is still an unresolved problem. As reported by Sekine [85], a fine-grained classification of entities scales up to hundreds of types. Moreover, as discussed in Section 2.5.3, significant differences may exist among individual entities or queries, even if the entities of the same class. For instance, a significant portion of users may anticipate the specific works rather than the generic intents with respect to the famous stars *Mozart* and *Avril*.

(3) As for query intent inference, many algorithms (e.g., [46, 73, 77]) have treated the whole query as the minimum analysis unit. They would suffer from the sparseness problem when processing previously unseen queries.

Different from the traditional wisdom of query parsing and understanding, we struc-

turally parse a user query by introducing two intent roles rather than a raw query segmentation. Compared with NERQ merely focusing on named entities, intent role annotation involves each composing unit inside a query and differentiates the utility of terms in expressing user's information need. Furthermore, kernel-object is not limited to named entities but a superset of name entity, which also avoids the requirement of a fine-grained classification of entities on the web. The kernel-object oriented query understanding (e.g., the concept of modifier graph in Chapter 6) achieves a per-kernel-object granularity. But it is not limited to a per-kernel-object level, the kernel-objects can also be further organized at a class-level if needed. As for Geo intent inference in Section 2.5.3, if we view Geo intent involving location terms as a geographic personalization, there can be a commercial personalization involving shopping terms or an academic personalization involving research terms, which corresponds to a fine-grained classification of modifiers that reflect users' intents. Thus, intent role oriented query parsing provides a more flexible scheme of query parsing and understanding.

Chapter 3

Intent Roles

3.1 Introduction

This chapter is devoted to parse user queries at a fine-grained term level. Some interesting issues are investigated, for instance,

(1) What is the common way that users conceive queries to manifest their intents or information needs ?

(2) Are their any differences among the terms selected by users when formulating search queries ?

In fact, the answers to these user-centric questions are important for achieving a better query parsing and understanding.

Query log of search engines records massive users' historical searching behaviors, including submitted queries, reformulated queries, clicked web pages, etc. The large amounts of query log available to us today necessitate a number of methods for interpreting users' search behaviors or improving retrieval performance. Inspired by the previous studies, we also resort to query log for knowledge to interpret users' search behaviors. In particular, we observe that: when formulating a query with respect to a particular intent, the selected terms play different roles in expressing this intent. Thus we propose two types of intent roles (kernel-object and modifier) to differentiate the utilities of terms within a query.

The rest of this chapter is organised as follows: Section 3.2 depicts the composing of a query log. Some query log oriented terminologies that are used throughout this thesis are summarized and/or defined. Moreover, we conducted a detailed analysis of the query

log SogouQ, which paves the way for out later SogouQ oriented experiments. In Section 3.3, we investigate users' search behavior in context of mul-session at a fine-grained term level, from which we derive the concepts of intent roles (kernel-object and modifier) to parse user queries. In particular, we discuss the potential values of intent role annotation in Section 3.4.

3.2 Query Log in Use

3.2.1 What A Query Log Looks Like

In real-life information search via a web search engine, from the initial query entered into the search box to the end that the user gets the desired information or gives up the search, this iterative process is entirely recorded. The resulting data is known as *query log* (or *clickthrough log*). We use the concept of query log across this thesis.

In general, a query log consists of massive records. Each record commonly includes the user identifier, timestamp, query, clicked web page, etc. The exact format of a record depends on the host web search engine. As an illustration, Figure 3.1 shows a fragment of the query log SogouQ (detailed in Section 3.2.3).

```
.....
7703190427880977    [网络游戏+冒险岛]    1 1    gamezone.qq.com/z/mxd/
13505142608748166   [gre]    5 2272  exam.21tx.com/fl/GRE/
6253342044314254    [钟丽缇]    3 3    www.pop99.com/Photo/mingx/z/200605/300.shtml
13505142608748166   [gre]    2 2273  edu.sina.com.cn/exam/gre/
4544542250785381    [rundll32.exe 下载]    14 6    www.ddduuu.com/IT/xitong/
.....
```

Figure 3.1: A fragment of the query log SogouQ.

As shown in Figure 3.1, each record in SogouQ consists of 4 fields: *user ID*, *query*, *rank*, *order*, *URL*. The meaning of each field is explained in Section 3.2.3.

3.2.2 Query Log Oriented Terminology

In the field of query log mining, some terminologies (e.g., session, reformulation, etc) are commonly used to model the composing data-parts in a query log and describe how a particular algorithm works. To get a clearer description of the query log based algorithms in this thesis, a number of terminologies are formally given as follows:

Definition 1 (Term) *Term (denoted as t) refers to a semantic unit when formulating a search query or a language sentence.*

For an English term, it may consist of multiple words. In the literatures [42, 69], the semantic unit that contains multiple words is called multi-word expression (MWE). For east asian languages (e.g., Chinese), a term consists of a sequence of characters.

Definition 2 (Query) *Query (denoted as q) refers to the text submitted by a user when searching information via a web search engine.*

A query usually consists of a sequence of terms. By the bivariate operator $\succ$, $t \succ q$ denotes that t is a composing term of query q.

Definition 3 (Query Log) *A query log is denoted as a set of records $\Gamma = \{< tp, q, u, CP >\}$, where tp is the time-stamp with respect to the record, q is the submitted query, u is an anonymous identifier of the original user, CP denotes the click-stamp, which may include the clicked URL and the rank of the clicked URL in the result list, etc.*

Given a query log Γ, we assume that $U = \{u\}$ denotes the set of all users and $Q = \{q\}$ denotes the set of all queries submitted by all users.

Definition 4 (Session) *A session is defined as a sequence of search behaviors performed by one particular user within a specific time interval. Formally, it is given as: $s = \{< tp, q^k, u_j, CP >\}$, the superscript k is used to specify that q^k is the k-th query in s.*

In the context of a session, if we merely care about the field of query in a record, a session can be simply denoted as a sequence of queries $s = \{q^1, ..., q^k\}$ due to the same user. We use this simple format of a session in this thesis. $|s|$ denotes the size of a session, i.e., the number of queries it contains, $q \in s$ denotes that query q appears in the session s, and $s \in \Gamma$ denotes a session in query log Γ.

Definition 5 (Sin-Session) *For the session that consists of a single search record, we call it a sin-session, denoted as* s^-.

Definition 6 (Mul-Session) *For the session that consists of multiple search records, we call it a mul-session, denoted as* s^+.

As for sin-session and mul-session, we also say that a sin-session or mul-session consists of a single query or multiple queries instead when we merely care about the field of query in a record.

Definition 7 (Reformulation) *Query* q_j *is said to be the reformulation of query* q_i*, if* q_j *follows* q_i *in a mul-session.*

Note that, q_i does not need to be the first query in a mul-session. In other literatures [73, 77], reformulation is also called *refinement*. The two terminologies are used equivalently in this thesis.

Definition 8 (Co-Session) *For two distinct queries* $q_i \in Q$ *and* $q_j \in Q$*, if* $\exists s \in \Gamma$ *that meets* $q_i \in s$ *and* $q_j \in s$*, we say* q_i *and* q_j *are co-session queries, denoted as* $CoSession(q_i, q_j)$.

Definition 9 (Co-Click) *Let* $D(q_i)$ *denote the clicked documents (or web pages) with respect to* q_i*, for two distinct queries* $q_i \in Q$ *and* $q_j \in Q$*, if* $D(q_i) \cap D(q_j) \neq \emptyset$*, we say* q_i *and* q_j *are co-click queries, denoted as* $CoClick(q_i, q_j)$.

Definition 10 (Co-Query) *For two distinct terms* t_i *and* t_j*, if* $\exists q \in Q$ *that meets* $t_i \succ q$ *and* $t_j \succ q$*, we say* t_i *and* t_j *are co-query terms, denoted as* $CoQuery(t_i, t_j)$.

Definition 11 (Term-Level Co-Session) *For two distinct terms* t_i *and* t_j*, if* $\exists q_m \in Q$ *and* $\exists q_n \in Q$ *that meet* $CoSession(q_m, q_n) \wedge t_i \succ q_m \wedge t_j \succ q_n$*, we say* t_i *and* t_j *are co-session terms, denoted as* $TCoSession(t_i, t_j)$.

Definition 12 (Term-Level Co-Click) *For two distinct terms* t_i *and* t_j*, if* $\exists q_m \in Q$ *and* $\exists q_n \in Q$ *that meet* $CoClick(q_m, q_n) \wedge t_i \succ q_m \wedge t_j \succ q_n$*, we say* t_i *and* t_j *are co-click terms, denoted as* $TCoClick(t_i, t_j)$.

As discussed in Section 2.5.2, there are a great number of literatures that derive knowledge from query log in different ways. For example, [38, 73, 77, 98] made use of the co-click, co-session information in a query log to cluster queries or differentiate the topics underlying a group of queries. [11, 13] built different types of graphs, e.g. query-flow graph, query-term graph, etc. Obviously, the co-session and co-click information are defined at a whole query level, while the co-query, term-level co-session and term-level co-click information are defined at a term level. In particular, we denote the statistics of co-session and co-click information derived from a query log as *query-level knowledge in query log (QKQL)*. In contrast, the statistics of co-query, term-level co-session and term-level co-click derived from a query log are denoted as *term-level knowledge in query log (TKQL)*. In fact, both QKQL and TKQL essentially are wisdom-of-crowds from massive users.

3.2.3 Query Log SogouQ

Public Query Log

Due to the privacy issue and the commercial importance, only a few query logs are publicly available. For example, the *AOL Query Log*[1](English), *MSN Query Log*[2](English), *SogouQ Query Log*[3](Chinese, two versions of 2008 & 2012) and *Yandex Query Log*[4](Russian). In this thesis, we focus on Chinese, thus SogouQ is adopted as our query log. Without specification, all the query log oriented experiments are based on SogouQ (2008-version).

The Nature of SogouQ

Currently, most query log oriented analytical studies are conducted using English query logs rather than Chinese query log. To get a general sense of the Chinese query log, we perform an analytical investigation with SogouQ, which also paves the way for our subsequent usage.

SogouQ spans a month (June in 2008) and contains about 30 million clicks. The meaning of each field within a record is shown in Table 3.1.

[1]http://www.gregsadetsky.com/aol-data/
[2]http://research.microsoft.com/en-us/um/people/nickcr/wscd09/
[3]http://www.sogou.com/labs/dl/q.html
[4]http://imat-relpred.yandex.ru/en/datasets

Field	Meaning
ID	Automatically assigned user id based on the Cookie information
Query	Search query submitted by a user
Rank	Rank of the clicked URL in the result list
Order	The click-order with respect to a click behavior
URL	The URL of a web page clicked by the user

Table 3.1: The meaning of each field in a record of SogouQ.

Before utilization, we filter the ill-formed records, e.g., the query that contains irregular characters, URL, etc. For SogouQ (2008-version), the timestamp of each record is not provided. We extract sessions as follows: The records of each day are grouped by automatically assigned user id in an order of the submitting sequence. The records with the same user id in a day are totally regarded to be one session. Table 3.2 shows the basic statistics of SogouQ after purification.

Item	Count
#queries	9,788,674
#unique query	2,856,130
#sin-session	4,909,197
#mul-session	1,687,767

Table 3.2: Basic statistics of SogouQ.

As shown in Table 3.2, the counts are at a tens of thousands scale. Thus, it is reasonable to believe that the statistical results based on SogouQ can reflect the characteristics of a common Chinese query log, and the experimental results are convinsible.

Figure 3.2 plots the query frequency distribution with respect to the query number, where the x-axes represents the number of queries with respect to a specific frequency, the y-axes represents query frequency.

As shown in Figure 3.2, a large majority of queries are repeated queries. The curve fits a Zipf[5] distribution well, which is consistent with the prior studies [33, 99].

Figure 3.3 plots the query length distribution. Different from English query, Chinese query is generally written with no term delimiters. The length here is roughly a character count.

As shown in Figure 3.3, the majority of queries are short ones. Through our calculation, the number of characters per query is 6.41 on average.

[5]http://en.wikipedia.org/wiki/Zipf's_law

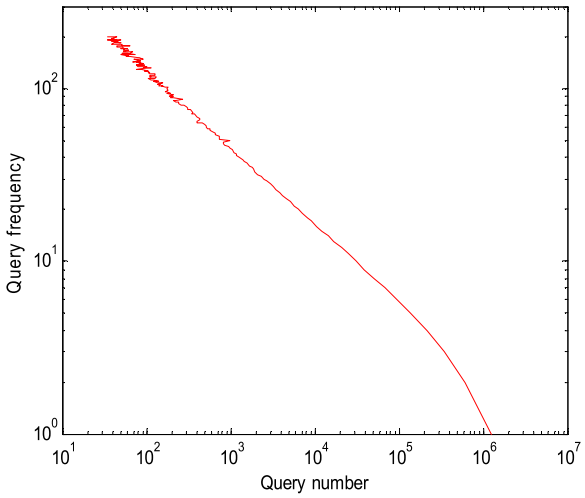

Figure 3.2: Query frequency vs. query number (log-log scale).

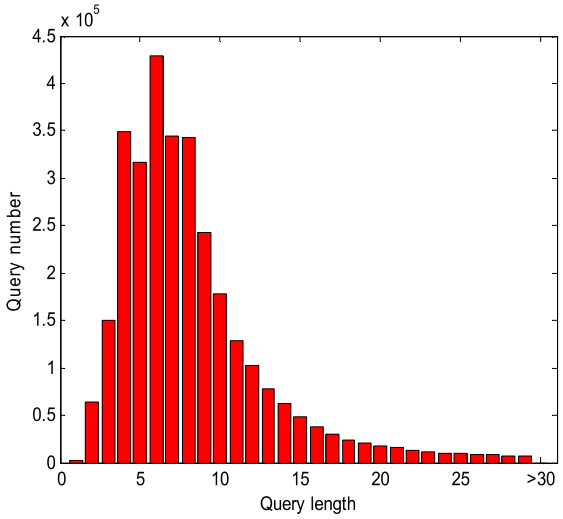

Figure 3.3: Query length distribution.

Figure 3.4 plots the session distribution, from where we can observe that: Sin-session (i.e., $|s| = 1$) is the most frequently occurring session, which approximately constitutes a ratio of 74% of all sessions.

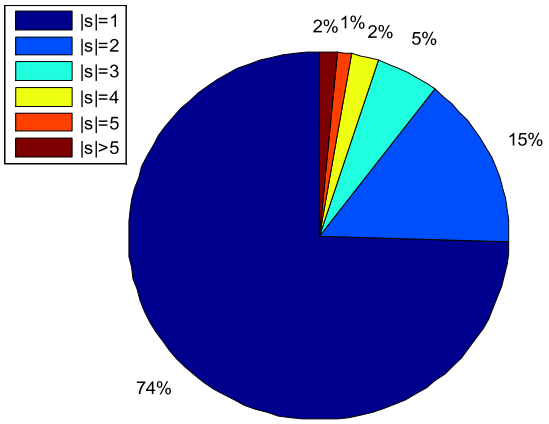

Figure 3.4: Session size distribution.

Going further, Figure 3.5 is a plot showing the average length of queries versus their position in the context of session. Specifically, the x-axis represents the position of a query inside a user session (i.e., the value of k in Definition 4), the y-axis represents the average length of queries in the same position.

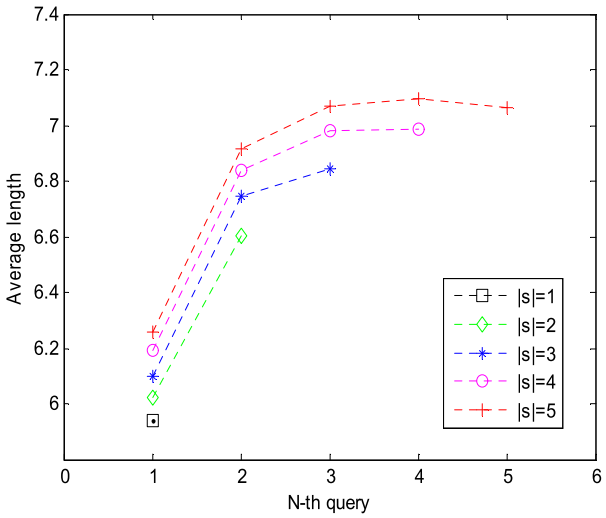

Figure 3.5: Distribution of query length in the context of session.

From Figure 3.5, an important characteristic of user session can be observed: The average length of the reformulated query is generally longer that the previous query, which reveals that the reformulated query expresses a more clear intent.

3.3 Intent Roles: Kernel-object and Modifier

Table 3.3 illustrates a session extracted from the query log SogouQ (the click information is omitted). Each query is segmented into terms, where "|" denotes a term delimiter.

NO.	Query	Composing Terms	Click
1	冠珠(Guanzhu)	冠珠(Guanzhu)	*
2	冠珠价格(Guanzhu price)	冠珠(Guanzhu) | 价格(price)	*
3	欧神诺瓷砖(Oceano ceramics)	欧神诺瓷砖(Oceano ceramics)	*
4	欧神诺瓷砖价格 (Oceano ceramics price)	欧神诺瓷砖(Oceano ceramics) | 价格(price)	*
5	瓷砖+价格+评价 (ceramics price review)	瓷砖(ceramics) | 价格(price) | 评价(review)	*
6	欧神诺瓷砖安徽价格 (Oceano ceramics Anhui price)	欧神诺瓷砖(Oceano ceramics) | 安徽(Anhui, a place) | 价格(price)	*
7	欧神诺瓷砖图片 (Oceano ceramics picture)	欧神诺瓷砖(Oceano ceramics) | 图片(picture)	*

Table 3.3: A session extracted from SogouQ.

Through examining the consecutive queries at a term granularity, we can easily figure out that the user performs several patterns of query reformulation to clarify his or her intents, such as term addition, term substitution, etc [47]. More specifically, these query reformulations are term-centered. For example, 冠珠(Guanzhu)-centered reformulations correspond to queries: 冠珠(Guanzhu) and 冠珠价格(Guanzhu price), which indicates that: 冠珠价格(Guanzhu price) should be decomposed into 冠珠(Guanzhu) and 价格(price), 价格(price) is a popular aspect of 冠珠(Guanzhu). Analogously, 欧神诺瓷砖(Oceano ceramics)-centered reformulations correspond to queries: 欧神诺瓷砖(Oceano ceramics), 欧神诺瓷砖价格(Oceano ceramics price), 欧神诺瓷砖安徽价格(Oceano ceramics Anhui price) and 欧神诺瓷砖图片(Oceano ceramics picture), which indirectly indicate the popular aspects: 价格(price) and 图片(picture). Moreover, we can deduce that: the centric terms, such as 冠珠(Guanzhu) and 欧神诺瓷砖(Oceano ceramics), play a dominant role that abstracts the core object of the user's intents. The co-appearing terms, such as 价格(price) and 图片(picture), play a role that specifies user's interested aspects. This observation also provides us the answer to the second question in Section 3.1, namely there are differences among the terms being selected to manifest a specific information need.

Inspired by the above observation, we propose two intent roles (*kernel-object* and

modifier) to interpret the different utilities of terms within a query as follows:

Definition 13 (Kernel-object) *Given a query with $n(n \geq 1)$ terms, kernel-object (denoted as ko) refers to the dominant term that abstracts the core object or topic expressed by this query.*

For example, the term 欧神诺瓷砖(Oceano ceramics) within the queries 欧神诺瓷砖价格(Oceano ceramics price) and 欧神诺瓷砖图片(Oceano ceramics picture). In this thesis, we assume that there is one and only one kernel-object inside a query. The study of queries including parallel concepts or more complicated cases is planed as our future work.

Definition 14 (Modifier) *For a query with $n(n \geq 1)$ terms, modifier (denoted as mo) refers to the co-appearing terms with respect to the kernel-object, which explicitly specifies user's interested attributes or concrete aspects.*

For example, the terms like 价格(price) and 图片(picture) in queries 欧神诺瓷砖价格(Oceano ceramics price) and 欧神诺瓷砖图片(Oceano ceramics picture). For the one term query, we assume that there is no modifier.

Essentially, kernel-object and modifier are labels used to differentiate terms in a query. But for convenience, sometimes we directly use kernel-object and modifier to refer to the terms annotated as kernel-object and modifier respectively. Based on the proposed intent roles, we further classify search queries into two classes.

Definition 15 (Role-explicit Query) *Given a query, if it can be represented with kernel-object and modifier, we call it a role-explicit query, denoted as q^+.*

For a role-explicit query, we assume that a composing term plays either the role of a kernel-object or the role of a modifier. Once the kernel-object is identified, the other co-appearing terms are regarded as modifiers. Without loss of generality, ignoring the order information, a role-explicit query can be represented as $q^+ = ko + \{mo\}$, i.e., a kernel-object plus a set of co-appearing modifiers.

Definition 16 (Role-implicit Query) *Given a query, if it can not be represented with kernel-object and modifier, we call it a role-implicit query, denoted as q^-.*

For instance, 萧潇出演过什么电视剧(the TV plays that XiaoXiao has starred in) and 培养孩子读书习惯(cultivate children's reading habits) are two role-implicit queries, these queries can't be well expressed with kernel-object and modifier.

Besides the definition of intent roles, how well this idea can work in reality is the subject of this thesis. Thus some fundamental problems for performing intent role oriented query parsing must be addressed, they are:

Problem-1: Given an arbitrary query, how to determine whether this query is a role-explicit query or a role-implicit query ?

Problem-2: Given a role-explicit query, how to identify its composing kernel-object and modifier ?

Problem-3: How well the idea of intent role oriented query parsing can work in reality ?

Corresponding to the Problem-1 and Problem-2, in Chapter 5, we study how to perform intent role annotation and role-explicit query identification, and specific algorithms are proposed. As for Problem-3, in Chapter 6, we propose the strategy of subtopic mining via modifier graph clustering, which is an extended application of intent role oriented query parsing.

3.4 Preliminary Discussion of Intent Role Annotation

In this section, we perform a preliminary discussion of intent role oriented parsing from different perspectives, namely a single-query perspective and an aggregative perspective.

3.4.1 A Single-query Perspective

As an example, Table 3.4 illustrates a number of annotated role-explicit queries.

Queries	Intent role annotation
小说+哈利波特+下载 (fiction Harry Potter download)	*ko*: 哈利波特(Harry Potter) *mo*: 小说(fiction) *mo*: 下载(download)
哈利波特影视歌曲 (Harry Potter movie song)	*ko*: 哈利波特(Harry Potter) *mo*: 影视(movie) *mo*: 歌曲(song)
哈利波特小游戏(Harry Potter game)	*ko*: 哈利波特(Harry Potter) *mo*: 小游戏(game)

Table 3.4: Annotated role-explicit queries.

From Table 3.4, we can observe that: 哈利波特小游戏(Harry Potter game) can be

represented as: *ko:* 哈利波特(Harry Potter) plus *mo:* 小游戏*(game)*. Moreover, we can deduce that the user's interest is restricted at the 小游戏(game) aspect of the entity 哈利波特(Harry Potter). Thus, annotating role-explicit queries with intent roles, on one hand, provides us a new view of query representation, on the other hand, the utilities of terms in expressing a particular information need are effectively revealed.

3.4.2 An Aggregative Perspective

Despite a structural parsing per query, we further conduct an analysis in an aggregative perspective given a group of annotated role-explicit queries. In particular, let $CoKO(ko)$ denote a set of role-explicit queries in a query log that share the same kernel-object (the queries in $CoKO(ko)$ are not necessarily distinct from each other). In Chapter 6, $CoKO(ko)$ is formally defined as *Co-kernel-object Elements*, which refers to subtopic strings other than mere queries. Here only queries are taken into account.

Figure 3.6 illustrates a directed graph derived from a set of $CoKO(ko)$s. Specifically, both purple circle and purple diamond represent a kernel-object. A purple circle denotes that the kernel-object has ever been submitted as a query. Both green circle and green diamond denote a modifier. The term concatenation corresponding to the path starting from a green circle to its nearest kernel-object has ever been submitted as a query. An edge (path) from a kernel-object to the satellite modifiers means a co-appearance in one query. The edge is directed from a more frequent modifier to a less frequent modifier. A directed edge between two kernel-objects means a co-appearance in one user session. The edge width is proportional to co-occurrence frequency of two linked terms.

As shown in Figure 3.6, the aggregated role-explicit queries performed by different users are meaningful. The aggregated graph can be interpreted as the wisdom of crowds on clarifying their intents. The role-explicit queries within $CoKO(ko)$ correspond to a star topology, which reflects a set of *ko*-oriented intents conceived by different users.

For example, for *ko*=哈利波特(Harry Potter), it has ever been submitted not only as a one-term query, but also as the kernel-object of many other queries, such as 哈利波特影视歌曲(Harry Potter movie song), 哈利波特小游戏(Harry Potter game), etc. For the same kernel-object, some users prefer information in the aspect of 小说(fiction), some users prefer information in the aspect of 游戏(game). While in the category of 影视(film), 歌

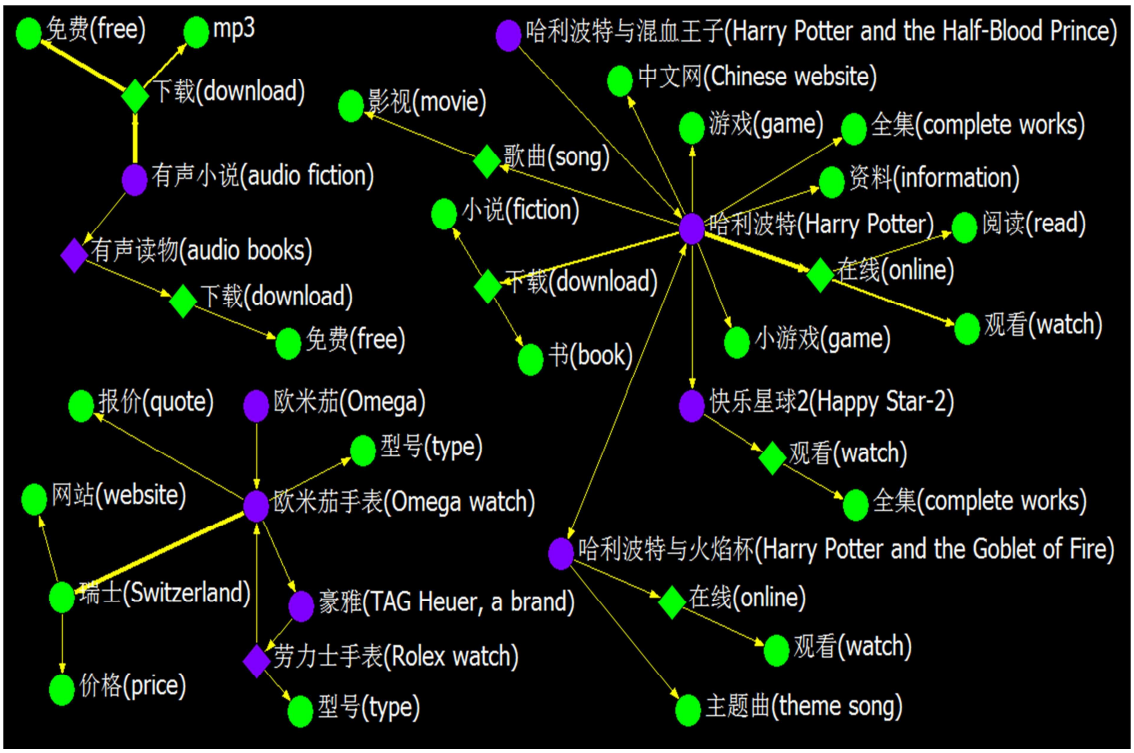

Figure 3.6: The directed graph derived from a set of $CoKO(ko)$s.

曲(song) and 电影(movie) are both popular aspects. Furthermore, the satellite modifiers of 哈利波特(Harry Potter) indicate that it would be highly underspecified and/or ambiguous if directly submitted as a query. Without any context or user's preference, it is impossible to know the exact intent encoded within query 哈利波特(Harry Potter). Unfortunately, 哈利波特(Harry Potter) was directly submitted as a one-term query with a total number of 649 times in our adopted query log. However, the aggregated modifiers provide us a way of how to capture the underlying intents of single-kernel-object queries like 哈利波特(Harry Potter). The edge between two kernel-objects implies that a user turns to a new kernel-object oriented intent after searching the previous kernel-object oriented intent, e.g., between 哈利波特(Harry Potter) and 快乐星球*2*(Happy Star-2). Analogously, the same situation holds true for other kernel-objects, e.g., 有声小说(audio fiction), 欧米茄手表(Omega watch), etc.

Besides query intent understanding, the modifiers that act as strong indicators of diverse intents are also useful for document ranking. When providing a search result list, a search engine must balance the intents of different users. So an effective organization of aggregated modifiers is also valuable for providing a diversified result to maximize the

probability that a user is satisfied. E.g., for underspecified query 哈利波特 (Harry Potter), the pre-mined modifiers 游戏(game), 小说(fiction) and 中文网(Chinese website) give us a hint of how to rank the related documents. It is natural that the result list, which matches a topical relevance from different aspects, e.g., 游戏(game), 小说(fiction) and 中文网(Chinese website), will enhance user experience than a result list that matches a single topical relevance. For a more explicit query 哈利波特在线(Harry Potter online), the modifiers 阅读(read) and 观看(watch) imply two lines of information needs desired by different users.

Moreover, if we view Geo intent (Section 2.5.3) involving location terms as a geographic personalization, there can be a commercial personalization involving shopping terms or an academic personalization involving research terms, which corresponds to a fine-grained classification of modifiers that reflect users' intents. Thus, it can be concluded that intent role annotation provides us a more flexible scheme of query parsing.

Chapter 4

Role-explicit Query Extraction

4.1 Introduction

In this chapter, we study how to extract role-explicit queries from a given query log. The value lies in that a repository of sufficient annotated role-explicit queries is a fundamental data source for deriving the model lexicon of the SWNR model (Chapter 5).

A straightforward strategy can be carried out as: Firstly segmenting the query dataset using a typical query segmentation approach. Then manually identify the role-explicit queries and label the intent roles at the same time. However, this strategy suffers from some serious limitations on the feasibility:

(1) Query segmentation is potentially ambiguous. An ambiguous segmentation would further result in an ambiguous annotation of intent roles.

(2) As we discussed later in this chapter, for some role-explicit queries, different users may have different opinions when annotating the kernel-object and modifier.

(3) A massive manual annotation is labor intensive.

In contrast, we propose an unsupervised system, which performs role-explicit query extraction concurrently with intent role annotation. The rest of this chapter is organized as follows: Section 4.2 formulates the problem of role-explicit query extraction. In Section 4.3, we detail the SWGV approach that performs mul-session based extraction. Section 4.4 details the CD approach that performs sin-session based extraction. The practical performance of the proposed system is evaluated through experiments on a real query log in Section 4.5.

4.2 Problem Formulation and System Architecture

Before formulating the problem of role-explicit query extraction, we give the definition of ko-centered reformulation, which based on the definition of formulation (Section 3.2.2) and intent roles (Section 3.3).

Definition 17 (Ko-centered Reformulation) *Query q_j is said to be a ko-centered reformulation of query q_i, if and only if:*

(1) query q_j is a reformulation of query q_i;

(2) query q_i and query q_j are role-explicit queries;

(3) the kernel-objects of query q_i and query q_j are the same;

For example, in a mul-session consisting of two role-explicit queries: 哈利波特(Harry Potter) and 哈利波特小说(Harry Potter fiction), 哈利波特小说(Harry Potter fiction) is a ko-centered reformulation of 哈利波特(Harry Potter), and they share the same kernel-object 哈利波特(Harry Potter).

The problem of role-explicit query extraction is a task defined as: Given the query log Γ, we wish to extract a repository of role-explicit queries $R = \{< q^+, ira(q^+) >\}$, where $ira(q^+)$ denotes the intent role annotation of role-explicit query q^+.

In this chapter, we propose a system that can automatically extract role-explicit queries with a high precision. The key idea of our system is that: For a specific query q, if we can determine a term $t \succ q$ as its kernel-object with a sufficient degree of certainty, it in turn implies that query q is a role-explicit query.

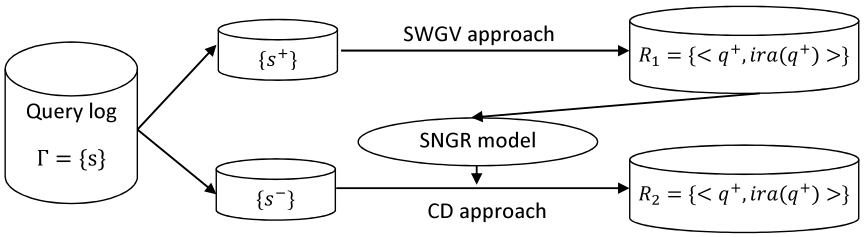

Figure 4.1: The system architecture of extracting role-explicit queries.

Figure 4.1 shows the architecture of the proposed system for extracting role-explicit queries. In particular, we separate all sessions $s \in \Gamma$ as mul-sessions $\{s^+\}$ and sin-sessions $\{s^-\}$. According to the session type, we devise different approaches to extract role-explicit queries. Namely, for mul-sessions, the *Sliding-window plus Global-voting approach*

(SWGV) (Section 4.3) is proposed, which consists of the *Sliding-window algorithm* (Section 4.3.2) and the *Global-voting algorithm* (Section 4.3.3). For sin-sessions, the *Combined-dictionary approach* (CD) (Section 4.4) is proposed, which is a combined utilization of the linguistic dictionary and the *simplified n-gram role* language model (Section 5.3). We detail how the proposed system works in the following sections.

4.3 SWGV Approach for Mul-sessions

4.3.1 Certainty of Being the Kernel-object

Table 4.1 shows the corresponding intent role annotation for queries in Table 3.3. For traditional query segmentation, 欧神诺瓷砖(Oceano ceramics) would be segmented at a smaller granularity, e.g., 欧神诺(Oceano) and 瓷砖(ceramics). However, in the context of a session, we find that the original user keeps the substring 欧神诺瓷砖(Oceano ceramics) invariable when reformulating queries to clarify his or her information need. The original user's reformulating behavior indirectly indicates that 欧神诺瓷砖(Oceano ceramics) should be treated as one semantic unit (e.g., a multi-word expression). This is also consistent with the study conducted by Du et al. [34], who showed that the invariable substring generally corresponds to a semantic unit. Thus, we prefer longer semantic units in this thesis.

NO.	Query	Intent role annotation
1	冠珠(Guanzhu)	*ko*: 冠珠(Guanzhu, a brand)
2	冠珠价格(Guanzhu price)	*ko*: 冠珠(Guanzhu) *mo*: 价格(price)
3	欧神诺瓷砖(Oceano ceramics)	*ko*: 欧神诺瓷砖(Oceano ceramics)
4	欧神诺瓷砖价格 (Oceano ceramics price)	*ko*: 欧神诺瓷砖(Oceano ceramics) *mo*: 价格(price)
5	瓷砖+价格+评价 (ceramics price review)	*ko*: 瓷砖(ceramics) *mo*: 价格(price) *mo*: 评价(review)
6	欧神诺瓷砖安徽价格 (Oceano ceramics Anhui price)	*ko*: 欧神诺瓷砖(Oceano ceramics) *mo*: 安徽(Anhui, a place) *mo*: 价格(price)
7	欧神诺瓷砖图片 (Oceano ceramics picture)	*ko*: 欧神诺瓷砖(Oceano ceramics) *mo*: 图片(picture)

Table 4.1: Intent role annotation of the queries in Table 3.3 (Section 3.3).

As shown in Table 4.1, once we annotate the queries with intent roles, we can observe that:

(1) Kernel-objects generally have a longer character length than co-appearing modifiers. The reason is that kernel-objects mainly are named entities, noun and multi-word expressions.

(2) Kernel-objects generally are the initially entered terms, e.g., 冠珠(Guanzhu) and 欧神诺瓷砖(Oceano ceramics).

(3) Users commonly perform ko-centered reformulations to clarify their information needs. E.g., 冠珠价格(Guanzhu price) is a ko-centered reformulation of 冠珠(Guanzhu). 欧神诺瓷砖价格(Oceano ceramics price) is a ko-centered reformulation of 欧神诺瓷砖(Oceano ceramics). From Table 4.1, we can deduce several patterns of ko-centered reformulation, such as modifier addition, modifier substitution, etc.

(4) The role-explicit queries in a mul-session are topically relevant, which can be justified by the fact that multiple role-explicit queries share the same kernel-object, which corresponds to an invariable constituent of a series of queries, e.g., 冠珠(Guanzhu) and 欧神诺瓷砖(Oceano ceramics). This phenomenon is particularly frequent for mul-sessions, which strongly indicates that the common substrings among consecutive queries are worth exploiting to facilitate the kernel-object identification.

Through capturing these intrinsic features observed above, we propose function $KORank()$ to quantify the certainty of a term being the kernel-object. Specifically, $KORank()$ is defined as:

$$KORank(t_k, q^i, s^+) = local(t_k, v_i) * global(t_k,\ s^+) \tag{4.1}$$

$$local(t_k, v_i) = e^{\frac{1}{|v_i|*k}} * \prod_{t_r \in v_i} \frac{|t_k|}{max_{t_j \in v_i}|t_j| - |t_r| + \alpha} \tag{4.2}$$

$$\begin{aligned} global(t_k, s^+) = & \frac{|\{q^j | q^j \in s^+, t_k \in q^j\}|}{|s^+|} \\ & * \frac{|\{< q^j, q^{j+1} > | < q^j, q^{j+1} > \in s^+, t_k \in < q^j, q^{j+1} >\}| + 0.5}{|s^+|} \\ & * \frac{1}{(max_{t_j \in v_i}|t_j| - |t_k| + 1)^2} \end{aligned} \tag{4.3}$$

where v_i represents the term vector of query q_i (q_i is the i-th query in the mul-session

s^+), t_k represents the k-th term in v_i, $|t_k|$ represents the length of t_k (i.e., the count of composing characters), $max_{t_j \in v_i}|t_j|$ represents the maximum term length in v_i, $|v_i|$ represents the number of terms in v_i, $|\{q^j|q^j \in s^+, t_k \in q^j\}|$ represents the number of queries including term t_k, $< q^j, q^{j+1} >$ represents a pair of consecutive queries in s^+, $t_k \in < q^j, q^{j+1} >$ means a consecutive appearance of t_k (namely both q^j and q^{j+1} include term t_k), $|\{< q^j, q^{j+1} > | < q^j, q^{j+1} > \in s^+, t_k \in < q^j, q^{j+1} >\}|$ represents the count of consecutive appearance for term t_k.

$local(t_k, v_i)$ combines the position and length features of a term. Instead of using the raw length of term t_k, we utilize the continued multiplication of the quotient between the length of term t_k and the difference of each term in v_i and the maximum term length. A constant α is added to the numerator (3 in our study), which is used as a factor to quantify the importance of term length. As the position of a term is inversely proportional to its weight of being the kernel-object, we take the exponent of the inverse value of the position of t_k, the size of v_i is used as a factor to discriminate different segmentations of the same query and biases the segmentation with fewer terms. The position of a term starts from 1. If we treat an individual query as a document, $local(t_k, v_i)$ can be interpreted as a local factor that quantifies t_k's superiority of representing the core topic of the intent encoded within query q^i.

$global(t_k, s^+)$ combines the session-level frequency and consecutive appearance count of term t_k. The raw frequency and consecutive appearance count are normalized by the total query number. Moreover, the square of the difference between the maximum term length and term t_k is used as a discount factor to bias long terms. To avoid the case that a zero consecutive appearance count results in a zero weight, a constant value is added (0.5 in our work). If we treat a mul-session consisting of multiple queries as a corpus, $global(t_k, s^+)$ can be viewed as a global factor that quantifies the topical stability of t_k among a mul-session.

Finally, $KORank()$ is defined as a product of $local(t_k, v_i)$ and $global(t_k, s^+)$ to quantify the certainty of $t_k \in v_i$ being the kernel-object. Different form TF-IDF model [81, 92] that the document frequency is inversely proportional to the term weight, our "document frequency" (term frequency and the consecutive appearance count) is directly proportional to the term weight, because the frequent term is more likely to be the kernel-object of a

series of ko-centered reformulations.

4.3.2 The Sliding-window Algorithm

Given the function $KORank()$, it can quantify the weight of being the kernel-object for any segment of a query in the context of a mul-session. Based on this function, we devise the sliding-window algorithm, which takes a mul-session as input and extracts a set of role-explicit queries (the intent roles are annotated at the same time) as output. Specifically, the sliding-window algorithm sequentially scans the queries in a mul-session with a sliding window, which encapsulates a sequence of three queries at a time. For each position of the sliding window, it is utilized to identify the kernel-object of the middle query. If a kernel-object is identified, this query will be viewed as a role-explicit query and its intent roles will be annotated correspondingly.

The reason why we set the window-size as three bases on the session distribution shown in Figure 3.4 of Section 3.2.3. Because the sessions are uneven distributed. The more queries a session includes, the lower ratio it constitutes. When the size of the sliding-window is set as 3, it can work with sessions with a size larger than 1. However, if we set the window-size as 5 (i.e., the previous two queries, the current query, the subsequent two queries), a large number of sessions can not meet this restriction (e.g., sessions of $|s|$=2 or $|s|$=3, which constitute a high ratio of the data set).

Figure 4.2 shows the scenario that the sliding-window slides to the i-th query q^i in mul-session s^+. We add two *null query* at the two ends to keep a consistency that both the initial query and the last query have a preceding query and/or a subsequent query.

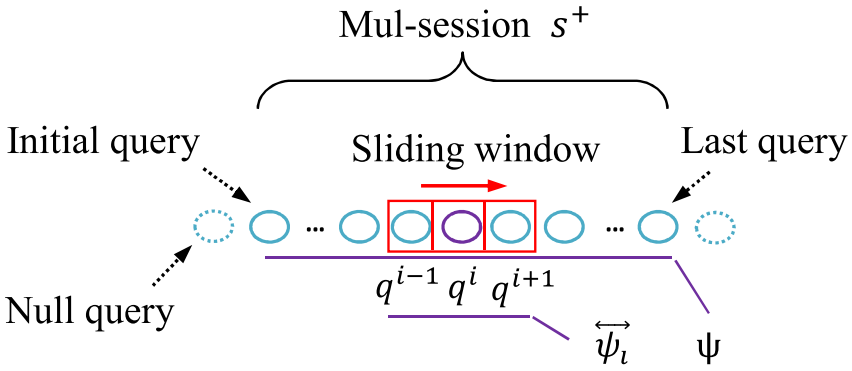

Figure 4.2: The sliding-window algorithm.

In particular, for the middle query, say q^i, in order to determine its kernel-object ko

if has, two kinds of kernel-object candidates are generated:

(1) *Dictionary oriented kernel-object candidate*: By performing adaptive query segmentation [104], we can get the term vector v_i of query q_i. In the context of mul-session s^+, each term $t_k \in v_i$ is computed a weight, i.e., $KORank(t_k, q^i, s^+)$. The term with the maximum weight is selected as a kernel-object candidate. We call it dictionary oriented kernel-object candidate, denoted as ko^+, i.e., $ko^+ = max_{t_k \in v_i} KORank(t_k, q^i, s^+)$.

(2) *Contextual information oriented kernel-object candidate*: For query q^i, we view the set of common substrings between q^i and its neighbor queries (q^{i-1} and q^{i+1}) as its contextual information, denoted as $\overleftrightarrow{\psi_i}$. To restrict the number of common substrings, only the longest common substrings are considered, i.e., $\overleftrightarrow{\psi_i} = \{lcs\}$, where lcs denotes a longest common substring. The punctuations within queries are considered as boundaries of longest common substrings. Moreover, all the longest common substrings between each pair of consecutive queries in the current mul-session are collected in ψ, which represents the session-level contextual information. We intuitively define several ad-hoc functions to distill kernel-object candidates from $\overleftrightarrow{\psi_i}$ as follows:

- $f_{ko}()$**:** Binary function $f_{ko}()$ indicates the association between a longest common substring lcs and the kernel-objects identified from the previous i-1 queries (denoted as $\overleftarrow{\Delta_{ko}}$), it is given as:

$$f_{ko}(lcs) = \begin{cases} 1 \; if \; lcs \in \overleftarrow{\Delta_{ko}} \\ 0 \; if \; lcs \notin \overleftarrow{\Delta_{ko}} \end{cases} \tag{4.4}$$

- $f_q()$**:** Binary function $f_q()$ indicates whether or not a lcs corresponds to a whole query of the current mul-session, it is given as:

$$f_q(lcs) = \begin{cases} 1 \; if \; lcs \in s^+ (i.e., lcs \; is \, a \, query) \\ 0 \; if \; lcs \notin s^+ \end{cases} \tag{4.5}$$

- $f_{mwe}()$**:** Based on the session level contextual information ψ, function $f_{mwe}()$ is defined to determine whether a lcs is a possible MWE that is not included in our

pre-established dictionary D. It is intuitively given as:

$$f_{mwe}(lcs) = \begin{cases} 1\ if\ lcs \notin D \wedge c(lcs, \psi) \geq 1\, when\, |s^+| = 2 \\ 1\ if\ lcs \notin D \wedge c(lcs, \psi) \geq 2\, when\, |s^+| \geq 2 \\ 0\ otherwise \end{cases} \tag{4.6}$$

where $lcs \notin D$ means that the lcs is not included by the pre-established dictionary D, $c(lcs, \psi)$ means the count of lcs in ψ.

- $f_{reg}()$**:** Through regular expression matching, binary function $f_{reg}()$ aims to capture the well-formed substrings like those in Table 4.2, which are hard to be recognized correctly.

哈利波特7 (Harry Potter 7)	诺基亚N95(Nokia N95)	卡巴斯基6.0 (Kaspersky 6.0)

Table 4.2: Well-formed substrings that should be viewed as semantic units.

$$f_{reg}(lcs) = \begin{cases} 1\ if\ lcs\ matches\, predefined\, regular\, expression \\ 0\ otherwise \end{cases} \tag{4.7}$$

Once a longest common substring $lcs \in \overleftrightarrow{\psi_i}$ matches a function defined above, we assume that this lcs is a well-formed semantic unit, and is viewed as a kernel-object candidate (denoted as cko). Otherwise, it will be directly discarded. Thus, corresponding to $\overleftrightarrow{\psi_i}$, a set of kernel-object candidates will be generated, denoted as $\psi_i^{cko} = \{cko\}$.

Algorithm 1 details the procedures of the sliding-window algorithm.

As shown in Algorithm 1, for the current query (say, q^i) to be processed, initially, we assume that it is a role-explicit query and try to determine its kernel-object ko. Finally, if we can obtain a reasonable term as its kernel-object, q^i will be extracted as a role-explicit query, otherwise it will be rejected as a role-implicit query.

In Step-3, the binary function $f_{ri}()$ aims to filter the obvious role-implicit queries that

Algorithm 1 The sliding-window algorithm: $R^{'} \leftarrow SWindow(s^{+})$.

Input: A mul-session, s^{+};
Output: A set of preliminarily annotated role-explicit queries, $R^{'} = \{< q, ira(q) >\}$

1: $\overleftarrow{\Delta_{ko}} = \phi$; $R^{'} = \phi$; initialize ψ;
2: **for** $1 \leq i \leq |s^{+}|$ **do**
3: **if** $f_{ri}(q^i) \vee |q^i| == 1$ **then**
4: continue;
5: **end if**
6: ko=null;
7: **if** $|q^i| \leq 3$ **then**
8: $ko = q^i$; $ira(q^i) = IRA(ko, q^i)$; $< q^i, ira(q^i) > \rightarrow R^{'}$; $ko \rightarrow \overleftarrow{\Delta_{ko}}$; continue;
9: **end if**
10: $ko^{+} = max_{t_k \in v_i} KORank(t_k, q^i, s^{+})$; initialize $\overleftrightarrow{\psi_i}$;
11: **for** each $lcs \in \overleftrightarrow{\psi_i}$ **do**
12: **if** $f_{ko}(lcs) \vee f_q(lcs) \vee f_{mwe}(lcs) \vee f_{reg}(lcs)$ **then**
13: $lcs \rightarrow \psi_i^{cko}$;
14: **end if**
15: **end for**
16: Sort ckoes in ψ_i^{cko} in order of their decreasing value $KORank(cko, q^i, s^{+})$;
17: **for** each $cko \in \psi_i^{cko}$ **do**
18: **if** $|cko| \geq |ko^{+}| \wedge |cko| \geq 2$ **then**
19: $ko = cko$; $ira(q^i) = IRA(ko, q^i)$; $< q^i, ira(q^i) > \rightarrow R^{'}$; $ko \rightarrow \overleftarrow{\Delta_{ko}}$; break;
20: **end if**
21: **end for**
22: **if** $null == ko \wedge |ko^{+}| \geq 2$ **then**
23: $ko = ko^{+}$; $ira(q^i) = IRA(ko, q^i)$; $< q^i, ira(q^i) > \rightarrow R^{'}$; $ko \rightarrow \overleftarrow{\Delta_{ko}}$;
24: **end if**
25: **end for**
26: **return** $R^{'}$

includes interrogative terms, it is given as:

$$f_{ri}(q) = \begin{cases} 1\ if\ q\ includes\, an\, interrogative\, term \\ 0\ otherwise \end{cases} \tag{4.8}$$

In particular, Table 4.3 shows the set of interrogative terms used in Equation 4.8.

吗(?) / 哪(where) / 谁(who) / 怎么(how) / 何时(when) / 多少(how many) / 怎样(how about) / 哪个(which) / 什么(what) / 如何(how) / 能否(can) / 什么是(what is) / 是否(whether)/ 为什么(why)

Table 4.3: The set of interrogative terms.

Steps 7 through 9 directly annotate q^i as a role-explicit query that consists of a single kernel-object if its length is shorter than 4. Because a Chinese term commonly consists of two or three characters, we view this kind of query as a one-term query. Given the identified kernel-object ko, function $IRA(ko, q^i)$ performs the corresponding intent role annotation $ira(q^i)$. Namely, set ko as the kernel-object of q^i, the remaining part of q^i will be segmented into a group of modifiers.

In step 10, we initialize the contextual information $\overleftrightarrow{\psi_i}$ and compute the dictionary oriented kernel-object candidate ko^+.

When the contextual information $\overleftrightarrow{\psi_i}$ is not empty (in steps 11 through 21), we firstly distill the well-formed longest common substrings through the predefined Functions ($f_{ko}()$, $f_q()$, $f_{mwe}()$, $f_{mwe}()$ and $f_{reg}()$) for generate kernel-object candidates $\psi_i^{cko} = \{cko\}$. For each candidate that derives from a longest subtopic string, we perform *cko-based segmentation*. Namely, the *cko* is directly viewed as a semantic unit of query q^i, the remaining parts are further segmented. The obtained term vector v_{cko} is used as the parameter to compute the weight of *cko* as $KORank(cko, q^i, s^+) = local(cko, v_{cko}) * global(cko, s^+)$. Then, these candidates are sorted in order of their descending value of $KORank(cko, q^i, s^+)$. We sequentially scan this ranked list, only the one meets the length condition that its character length is not shorter than the dictionary oriented kernel-object candidate ko^+ will be accepted. If no one meets the length condition, the ko^+ will be directly selected as the kernel-object when it has a length larger than 1. After parsing q^i, we slide the window down one query.

4.3.3 The Global-voting Algorithm

By the sliding-window algorithm, we can perform role-explicit query extraction per mul-session. However, the same query can be submitted by different users. A specific query $q \in Q$ may appear in different mul-sessions. Due to different contextual information in different mul-sessions, the same query may be identified as a role-explicit query but with different intent role annotations. If we view an annotation derived from a mul-session as a "vote" from the original user, it is reasonable to prefer the one that users are most likely to "vote for".

Towards this direction, we propose the *global-voting algorithm*. Specifically, for query q, let $R_q = \{ira(q)\}$ be the set of all intent role annotations for q derived from the mul-sessions using the sliding-window algorithm, then $ira(q)^* = max_{ira(q) \in R_q} c(ira(q), R_q)$ corresponds to the annotation that users are most likely to "vote for", where $c(ira(q), R_q)$ means the count of $ira(q)$ in R_q.

4.3.4 SWGV Approach

The sliding-window algorithm performs role-explicit query extraction and intent role annotation per mul-session, while the global-voting algorithm performs annotation selection leveraging on the wisdom of crowds. By combining the sliding-window algorithm and the global-voting algorithm, we get the the *Sliding-Window plus Global Voting* (denoted as SWGV) approach as shown in Algorithm 2.

As shown in Algorithm 2, we only consider the queries that appear at least two times (i.e., $c(q, Q) \geq 2$ in Step 10). And the annotation $ira(q)^*$ that gets the maximum votes is accepted as the final intent role annotation. The threshold 1 ensures that: even for a query that appears two times, the accepted annotation gets at least two "votes".

4.4 CD Approach for Sin-sessions

For queries in sin-sessions that have no contextual information, we resort to the linguistic linguistic dictionary based method and the *simplified n-gram role* language model (SNGR) proposed in Section 5.3. Because the SNGR model essentially works based on a model lexicon, thus we call the combined approach as *Combined-Dictionary* (CD)

Algorithm 2 Sketch of the SWGV approach.

Input: Mul-sessions of a query log Γ;
Output: A set of annotated role-explicit queries, $R_1 = \{< q, ira(q) >\}$;

```
1: R_1 = φ; Y = φ;
2: for each s^+ ∈ Γ do
3:    R' = SWindow(s^+);
4:    for each < q, ira(q) >∈ R' do
5:       ira(q) → R_q; update < q, R_q > in Y;
6:    end for
7: end for
8: for each < q, R_q >∈ Y do
9:    ira(q)* = max_{ira(q)∈R_q} c(ira(q), R_q);
10:   if c(q, Q) ≥ 2 ∧ c(ira(q)*, R_q) > 1 then
11:      < q, ira(q)* >→ R_1;
12:   end if
13: end for
14: return R_1
```

approach.

Specifically, for a query q, we first segment it to get its term vector v using the adaptive segmentation strategy [104] with the linguistic dictionary $\mathbb{D}$, then the term $t^* \in q$ that achieves $t^* = max_{t_k \in v} local(t_k, v)$ is selected as the dictionary oriented kernel-object candiate ko^+. Different from the way of using function $KORank()$ in Section 4.3.2, this time only the factor of $local(t_k, v_i)$ is used due to no context information.

We learn the model lexicon θ for the SNGR model using the role-explicit queries extracted from mul-sessions (using the SWGV approach discussed in Section 4.3.4). For a role-explicit query q^+, the optimal intent role annotation $ira(q^+)^*$ under the SWNR model is given as:

$$ira(q^+)^* = argmax_{ira(q^+) \in \Omega} p(ira(q^+)|q^+, \theta) \tag{4.9}$$

where Ω denotes the set of all possible intent role annotations for q^+.

For query q in a sin-session, if we initially assume it is a role-explicit query, under the SNGR model, we can obtain the intent role annotation $ira(q)^*$ that achieves the maximum likelihood. The kernel-object in $ira(q)^*$ is also considered as a kernel-object candidate, denoted as ko^-. Through a comparison between ko^+ and ko^-, we determine the most reasonable kernel-object ko for q.

Algorithm 3 shows how the CD approach operates.

Algorithm 3 Sketch of the CD approach.

Input: Sin-sessions of the query log Γ;
Output: A set of annotated role-explicit queries, $R_2 = \{< q, ira(q) >\}$;

1: $R_2 = \phi$;
2: **for** each $s^- \in \Gamma$ **do**
3: Initialize q with the single query in s^-
4: **if** $f_{ri}(q) \vee c(q, Q) < 3 \vee |q| == 1$ **then**
5: continue;
6: **end if**
7: **if** q is a new query **then**
8: $ko = null$;;
9: **if** $|q| \leq 3$ **then**
10: $ko = q$; $ira(q) = IRA(ko, q)$; $< q, ira(q) > \rightarrow R_2$; continue;
11: **end if**
12: $ko^+ = max_{t_k \in v} local(t_k, v)$; $ko^- \leftarrow argmax_{ira(q) \in \Omega} p(ira(q)|q, \theta)$;
13: **if** $|ko^-| \geq |ko^+| \wedge |ko^-| \geq 2$ **then**
14: $ko = ko^-$; $ira(q) = IRA(ko, q)$; $< q, ira(q) > \rightarrow R_2$;
15: **else if** $|ko^+| \geq 2$ **then**
16: $ko = ko^+$; $ira(q) = IRA(ko, q)$; $< q, ira(q) > \rightarrow R_2$;
17: **end if**
18: **end if**
19: **end for**
20: **return** R_2

In Algorithm 3, we only consider the queries that appear at least three times(i.e., $c(q, Q) \geq 3$). For the same query in different sin-sessions, the CD approach generates the same result. So in Step 7, the queries that have been processed will be skipped directly (just increase its count if needed). Similar to the sliding-window algorithm in Section 4.3.2, in steps 9 through 11, query q will be viewed as a role-explicit query that consists of a single kernel-object if its character length is shorter than 4. In Step 12, we compute the dictionary oriented kernel-object candidate ko^+ with no contextual information (merely the function of $local()$ in Section 4.3.1) and the kernel-object candidate ko^- determined by the SNGR model, which is learned using the role-explicit queries extracted from mul-sessions. In Step 14, if the character length of ko^- is not shorter than ko^+ and is larger than 1 at the same time, it will be accepted as the final kernel-object. Otherwise, ko^+ will be selected as the final kernel-object if its length is larger than 1.

4.5 Performance Evaluation

4.5.1 Data

The pre-established dictionary $\mathbb{D}$ in our study is an integration of three linguistic resources:

(1) **CEDICT**[1]: a Chinese-English bilingual dictionary. We integrate the Chinese entries (a total of 103,723) into our dictionary.

(2) **Wikipedia**[2]: a well-known online encyclopedia. We use the Chinese-version Wikipedia and all the entries (a total of 615,112) are integrated.

(3) **Baidu-Baike**[3]: the largest online Chinese encyclopedia and contains millions of entries. We collect the entries through two months' crawling. The number of crawled entries is 2,149,570.

By aggregating multiple linguistic resources, the obtained dictionary can provide an extremely wide coverage of named entities and specialized concepts, which helps us to cope with the heterogeneous queries.

4.5.2 Evaluation Metric

Inspired by metrics proposed for query segmentation (Section 2.2.4), we evaluate the performance of role-explicit query extraction at term-level and query-level respectively.

- **Term Level**: By treating the role-explicit queries as a set of terms with intent roles, the term-level metric aims to measure how well the automatically annotated queries recover these standard annotation (e.g., by human assessors). Let Θ be the standard annotation of role-explicit queries, $\Theta^{'}$ be the set of computed annotations, $komo()$ be the composing kernel-objects and modifiers. A kernel-object or modifier is viewed as "relevant" if and only if its intent role is correctly annotated. The widely used measures precision, recall and f-measure can be analogously defined as:

$$T_{pre} = \frac{komo(\Theta) \cap komo(\Theta^{'})}{komo(\Theta^{'})} \tag{4.10}$$

[1]http://www.mdbg.net/chindict/chindict.php
[2]http://en.wikipedia.org/wiki/Wikipedia:Database_download
[3]http://baike.baidu.com/

$$T_{rec} = \frac{komo(\Theta) \cap komo(\Theta^{'})}{komo(\Theta)} \tag{4.11}$$

$$T_{f-score} = \frac{2 * T_{pre} * T_{rec}}{T_{pre} + T_{rec}} \tag{4.12}$$

- **Query Level**: The query level metric becomes more rigid in judging how well the computed annotation matches the standard annotation. An annotation is viewed as correct if and only if the segmentation of the query and the intent role annotation of each term are all the same as the standard annotation. Query level precision Q_{pre} is the fraction of correctly annotated queries over the standard annotations:

$$Q_{pre} = \frac{\Theta \cap \Theta^{'}}{\Theta} \tag{4.13}$$

4.5.3 Test Collection and Baseline

Because the SWGV approach performs a global-voting, all the mul-sessions in SogouQ are used as an input to extract role-explicit queries. Finally, the obtained dataset $R_1 = \{< q, ira(q) >\}$ consists of 185,388 distinct instances. For sin-session based extraction, the sin-sessions in the 5th day of SogouQ are adopted to conduct the evaluation, the obtained dataset $R_2^{'} = \{< q, ira(q) >\}$ consists of 27,123 instances, which is a subset of target dataset $R_2 = \{< q, ira(q) >\}$. Based on R_1, we construct a standard collection (denoted as Δ_{mul}) for testing mul-session based extraction as follows: We randomly extract 1000 instances from R_1, and sequentially remove the noisy ones (e.g., salacious queries, English queries). For the top-500 accepted instance, we identify whether it is role-explicit or not, for a role-explicit we manually annotate its intent roles. Similarly, based on $R_2^{'}$, we construct another standard collection (denoted as Δ_{sin}) for testing sin-session based extraction.

Table 4.4 shows the statistics of the two test collections.

Collection	Size	Role-explicit query	Role-implicit query
Δ_{mul}	500	495	5
Δ_{sin}	500	492	8

Table 4.4: Test collections for evaluating role-explicit query extraction.

Because we assume that a query is role-explicit when we can determine a term as its kernel-object with a sufficient degree of certainty, so there are falsely extracted role-explicit queries in the test collection, as well as the system outputs. During evaluation, the falsely extracted role-explicit queries are computed as false annotations.

To our knowledge, there is no prior work conducted for role-explicit query extraction. To proceed with evaluation, we adopt the dictionary based kernel-object identification as the baseline approach. For a test instance q, its kernel-object is determined as $ko = max_{t_k \in v} local(t_k, v)$, the same as the dictionary oriented kernel-object candidate in Section 4.4. Thus the intent roles are annotated as $ira(q) = IRA(ko, q)$, which represents a single utilization of the linguistic dictionary.

4.5.4 Experimental Results

For mul-session based extraction, Table 4.5 shows the performance of the SWGV approach against the baseline approach on Δ_{mul}.

Approach	T_{pre}	T_{rec}	$T_{f-score}$	Q_{pre}
Baseline	70.25%	80.38%	74.97%	72.20%
SWGV	86.95%	91.66%	89.24%	88.00%

Table 4.5: Δ_{mul} based evaluation.

As shown in Table 4.5, the SWGV approach outperforms the baseline a lot. The reason for its high performance is its smart utilization of the contextual information and the global voting across all mul-sessions. The contextual information in a single mul-session helps to determine the kernel-object from the original user's perspective. The global voting further guarantees a more reasonable intent role annotation. As an illustration, for query 世界名著读后感(world masterpiece review), Table 4.6 shows the different annotations derived from different mul-sessions and their user votes.

Votes	Intent role annotation
29	*ko*: 世界名著(world masterpiece) *mo*: 读后感(review)
10	*ko*: 读后感(review) *mo*: 世界名著(world masterpiece)
6	*ko*: 世界名著读后感(world masterpiece review)
6	*ko*: 名著读后感(masterpiece review) *mo*: 世界(world)

Table 4.6: "Votes" for different intent role annotations.

From Table 4.6, we can observe that different annotations can be generated due to

different contextual information in mul-sessions. E.g., 世界名著(world masterpiece), 读后感(review) are annotated as the kernel-object respectively. However, the more reasonable kernel-object 世界名著(world masterpiece) wins the most votes, which demonstrates the effectiveness of the global-voting algorithm. Indeed, the other annotations are also meaningful and indicate different user's information needs. We leave the case of one query with multiple acceptable annotations as future work. The baseline approach merely relies on the linguistic dictionary. The drawback is that it works well for queries including named entities (on the condition that the named entities are included in the dictionary). E.g., for 哈利波特游戏(Harry Potter game), it can successfully determine 哈利波特(Harry Potter) as the kernel-object. However, it fails for the case of 哈利波特7下载(Harry Potter 7 download), for which 哈利波特7(Harry Potter 7) is more reasonable to be the kernel-object.

For sin-session based extraction, Table 4.7 shows the performance of the CD approach against the baseline approach on Δ_{sin}.

Approach	T_{pre}	T_{rec}	$T_{f-score}$	Q_{pre}
Baseline	60.99%	74.23%	66.96%	69.20%
CD	77.88%	81.37%	79.59%	81.60%

Table 4.7: Δ_{sin} based evaluation.

Due to the feature of independently parsing a query, the baseline approach shows slightly different performances in Table 4.5 and Table 4.7. The CD approach also outperforms the baseline approach. Despite the linguistic dictionary, the CD approach introduces the SNGR model to facilitate role-explicit query extraction. Because the training corpus for the SNGR model is the role-explicit queries extracted from mul-sessions, so SNGR can be interpreted as a statistical utilization of the user votes aggregated from mul-sessions. Therefore, the CD approach achieves a comparable performance to the SWGV approach. Put another way, we translate the contextual information and the global voting across mul-sessions as human wisdom to facilitate role-explicit query extraction. The experimental results in Table 4.5 and Table 4.7 show that our system is clearly favored by the indirect human wisdom and achieves a high precision.

Going further, we explore the recall of the proposed approaches, which can be interpreted as the ability of extracting positive role-explicit query instances out of the given query log. When performing the proposed approaches SWGV and CD, the following values

are counted:

(1) *Raw query*, which refers to the original queries of the source data.

(2) *Candidate query*, which refers to the valid query to be identified.

(3) *Extracted query*, which refers to the extracted role-explicit queries.

Table 4.8 and Table 4.9 show the statistical results when operating SWGV and CD approaches respectively.

Raw query	Candidate query	Extracted query	Extraction ratio
4,879,482	4,674,111	2,800,803	57.40%

Table 4.8: Statistical results when running SWGV approach.

Raw query	Candidate query	Extracted query	Extraction ratio
65,542	64,124	63,795	97.33%

Table 4.9: Statistical results when running CD approach.

With the availability of large scale query logs providing sufficient events, we would prefer precision than recall. The high extraction ratios in Table 4.8 and Table 4.9 show that the majority of role-explicit queries are extracted, so we can say that the proposed system also performs a robust role-explicit extraction.

Chapter 5

Intent Role Annotation and Role-explicit Query Identification

5.1 Introduction

In this chapter, we investigate the two fundamental problems of intent role oriented query parsing, i.e., intent role annotation and role-explicit query identification.

As for intent role annotation, a traditional way is the supervised method, like part-of-speech tagging [39, 95] and named entity recognition [35, 64]. However, these traditional methods for common natural language processing will suffer from some serious limitations when directly adopted for query processing:

(1) Query segmentation is potentially ambiguous. An ambiguous segmentation would further result in an ambiguous annotation of intent roles.

(2) In a query log, there are a large ratio of tail queries, which are hard to cope with in a supervised manner.

(3) A massive manual annotation is labor intensive.

Therefore, we resort to the statistical model, i.e., language modeling for help.

While for role-explicit query identification, we find that role-explicit queries can be differentiated from role-implicit queries by a set of predictive features. Thus, we resort to machine learning classifiers for solving this identification problem.

The rest of this chapter is organized as follows: Section 5.2 formally defines the two problems of intent role annotation and role-explicit query identification. In Section 5.3,

we present the simplified n-gram role language model for solving the problem of intent role annotation. In Section 5.4, based on a set of predictive features, we propose to address the problem of role-explicit query identification in a supervised manner. In Section In Section 5.5, a series of experiments are conducted to evaluate the proposed approaches.

5.2 Problem Formulation

5.2.1 The Task of Intent Role Annotation

Given a role-explicit query q^+, intent role annotation (IRA) refers to the task of annotating the intent role of each term $t \succ q^+$. For instance, for the role-explicit query *Harry Potter game*, the corresponding intent role annotation is given as:

$$\begin{array}{cc} \underline{\text{Harry Potter}} & \underline{\text{game}} \\ | & | \\ ko & mo \end{array}$$

where *Harry Potter* is annotated as the kernel-object, *game* is annotated as the modifier.

Essentially, intent role annotation involves two joint subtasks: term boundary detection and intent role labeling. Term boundary detection aims at properly segmenting a role-explicit query into a number of semantic units. Intent role labeling aims at correctly annotating the intent role of each term: either a kernel-object or a modifier.

As can be observed in query logs, some terms, such as 下载(download), 型号(type) and 评价(review), have a larger probability to be submitted as a modifier rather than a kernel-object. Some terms, such as 哈利波特(Harry Potter), 有声小说(audio fiction), 养老金(retirement pension) or other entities, have a larger probability to be submitted as a kernel-object and rarely occur as a modifier. While other terms, for instance, 电影(film), sometimes it plays the role of a kernel-object, e.g., in query 电影下载(film download), sometimes it plays the role of a modifier, e.g., in query 电影霸王别姬(film farewell my concubine). Given a set of role-explicit queries, once we decompose each query into a kernel-object and a set of modifiers, we would obtain a term vocabulary of kernel-objects and modifiers. This observation leads us to hypothesize that: There exists a latent model Ψ that describes how a specific role-explicit query is generated from a finite vocabulary

of kernel-objects and modifiers. Thus the discovery of Ψ helps us to address the problem of intent role annotation. In Section 5.3, the simplified n-gram role model is proposed to perform intent role annotation, which can be viewed as an approximation of Ψ.

5.2.2 The Task of Role-explicit Query Identification

Given an arbitrary query q, role-explicit query identification (RQI) refers to the task of identifying whether or not q is a role-explicit query, i.e., whether query q can be decomposed into a kernel-object and a set of modifiers.

By hypothesizing that role-explicit queries can be differentiated from role-implicit queries by a set of predictive features, we treat the task of role-explicit query identification as a binary classification problem. In Section 5.4, a number of machine learning classifiers are investigated to perform role-explicit query identification based on a set of predictive features.

5.3 Intent Role Annotation

5.3.1 Language Modeling

As a statistical technique, language model is widely used in many domains, such as speech recognition [75], machine translation [57], spell correction [54], information retrieval [70, 88], etc. In the context of information retrieval, the language model can be formulated as a probability distribution over string q, which reflects the likelihood of observing the string q.

The n-gram (NG) language model is widely used in language modeling, it assumes that the probability of observing the i-th word w_i in the context history of the preceding $i-1$ words depends on the shortened context history of the preceding $n-1$ words [22], which is expressed as:

$$p(w_i|w_1...w_{i-1}) = p(w_i|w_{i-n+1}...w_{i-1}) \tag{5.1}$$

And the probability of observing query $q = w_1...w_m$ is given as:

$$p(q = w_1...w_m) = p(w_m|w_1^{m-1})...p(w_2|w_1^1)p(w_1) = \prod_{i=1}^{m} p(w_i|w_{i-n+1}^{i-1}) \tag{5.2}$$

where w_i^j denotes the sequential words $w_i...w_j$. For the vocabulary $\mathbb{V}$ with $|\mathbb{V}|$ words, the number of parameters for the n-gram language model will be $|\mathbb{V}|^n$.

Brown et al. [16] proposed the n-gram class (NGC) language model to address the drawback of the n-gram language model that a huge number of parameters are required to be estimated. In the n-gram class language model, each word is associated to a unique class $\pi(w_i) = c_i$ and π is a function that maps a word into its class c_i (a total of $|\mathbb{C}|$ classes). The conditional probability is given by:

$$p(w_i|w_{i-n+1}^{i-1}) = p(w_i|c_i)p(c_i|c_{i-n+1}^{i-1}) \tag{5.3}$$

where c_i represents the word class to which the word w_i belongs. c_i^j represents the sequential classes $c_i...c_j$ corresponding to the sequential words $w_i...w_j$. For the vocabulary $\mathbb{V}$ with $|\mathbb{V}|$ words, the number of parameters of the n-gram class language model is decreased from $|\mathbb{V}|^n$ to $|\mathbb{C}|^n + |\mathbb{V}|$.

5.3.2 Language Model Encapsulating Intent Roles

Different from the n-gram language model and the n-gram class language model that work a word granularity, we perform language modeling for queries at a term granularity. Moreover, the intent role of each term is taken into account.

In particular, for a sequence of terms, say, $t_1...t_m$, let r_i^b ($b \in \{0, 1\}$) denote the intent role of the i-th term t_i, r_i^0 denote that t_i is a modifier, r_i^1 denote that t_i a kernel-object. Analogous to Equation 5.3, the probability of observing the i-th term t_i in the context history of the preceding $i-1$ terms can be given as:

$$p(t_i|t_1^{i-1}) = p(t_i|t_{i-n+1}^{i-1}) = \sum_b \{p(t_i|r_i^b)p(r_i^b|t_{i-n+1}^{i-1})\} \tag{5.4}$$

where only the shortened context history of the preceding $n-1$ terms is considered. Thus,

the probability of observing $t_1...t_m$ can be given as:

$$p(t_1...t_m) = p(t_m|t_1^{m-1})...p(t_2|t_1^1)p(t_1) = \prod_{i=1}^{m}\{\sum_b p(t_i|r_i^b)p(r_i^b|t_{i-n+1}^{i-1})\} \tag{5.5}$$

We denote the model defined by Equation 5.5 as *n-gram role language model* (NGR), which can be viewed as an extended version of the n-gram language model.

Specifically, we illustrate the topological differences among the n-gram language model, n-gram class language model and n-gram role language model in Figures 5.1, 5.2 and 5.3, where the red grid circles the conditional elements (e.g., words or terms). In the n-gram language model (Figure 5.1), the conditional probability depends on the preceding $n-1$ words.

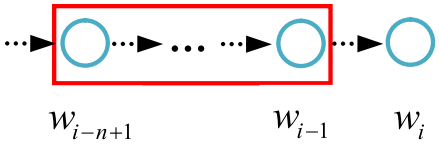

Figure 5.1: N-gram language model.

As shown in Figure 5.2, in the n-gram class language model, the probability of observing the i-th word w_i depends on the class attributes of the preceding $n-1$ words and the class attribute of word w_i.

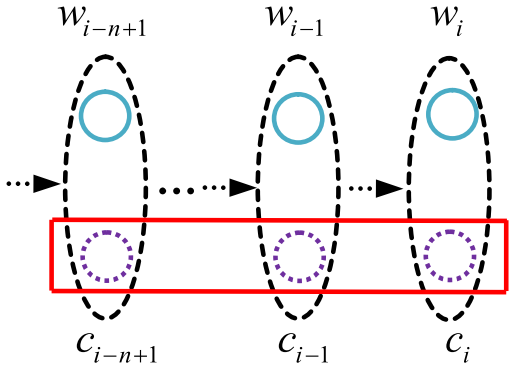

Figure 5.2: N-gram class language model.

While in the n-gram role language model (Figure 5.3), the probability of observing the i-th term t_i depends on not only the preceding $n-1$ terms but also the intent roles of the preceding $n-1$ terms and the intent role of term t_i.

Similar to the n-gram language model, the n-gram role language model also suffers from the drawback that a huge number of parameters are required to be estimated. For the

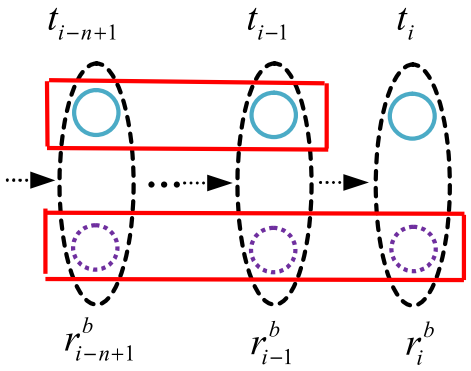

Figure 5.3: N-gram role language model.

vocabulary $\mathbb{V}$ with $|\mathbb{V}|$ terms the intent role of which are annotated, the parameters to be estimated will be as many as $(2|\mathbb{V}|)^n$. Note that, due to the assumption that there is one and only kernel-object inside a role-explicit query, the invalid annotations (e.g., an annotation that consist of two kernel-objects) will be ignored when using the NGR to model a role-explicit query.

Going further, we introduce three assumptions on how users conceive a role-explicit query as follows:

- **Assumption-1: Bag of kernel-object and modifier**: Under this assumption, a role-explicit query is represented as a bag of unordered kernel-object and modifiers.
- **Assumption-2: (Kernel-object)-dependent**: According to the nature of kernel-object and modifier (i.e., kernel-object abstracts the core object or topic of an information need encoded in a query, while modifier is used to specify the kernel-object), the intuition underlying this assumption is that: users first conceive the kernel-object, the co-appearing modifiers are conceived subsequently and (kernel-object)-dependent.
- **Assumption-3: Modifier mutually independent**: For a role-explicit query that contains multiple modifiers, this assumption supposes that the modifiers are mutually independent.

The union of the Assumptions 1, 2 and 3 depicts the process of how a user formulates a role-explicit query like that: The user first conceives the kernel-object that abstracts the core object or topic of his/her information need or intent, then a set of mutually independent

modifiers are conceived subsequently to specify the interested aspects if needed. This also provides the answer to the question in Section 3.1, i.e., What is the common way that users conceive queries to manifest their intents or information needs.

Suppose the intent role of the role-explicit query q^+ has been annotated, say, $q^+ = t_1^- ... t_{k-1}^- t_k^+ t_{k+1}^- ... t_m^-$, where t_i^+ denotes that term t_i is the kernel-object, while t_i^- denotes that term t^i is a modifier. Under the Assumptions 1, 2 and 3, the probability of observing q^+ can be given as:

$$\begin{aligned} p(q^+) &= p(t_1^- ... t_{k-1}^- t_{k+1}^- ... t_m^- | t_k^+) p(t_k^+) \quad //Under Assumption-1 \ \& \ Assumption-2 \\ &= p(t_k^+) \prod_{i:i \neq k} p(t_i^- | t_k^+) \qquad //Under Assumption-3 \end{aligned} \tag{5.6}$$

We call the model defined by Equation 5.6 as *simplified n-gram role* (SNGR) language model. Figure 5.4 illustrates its topological structure, which breaks down the context history.

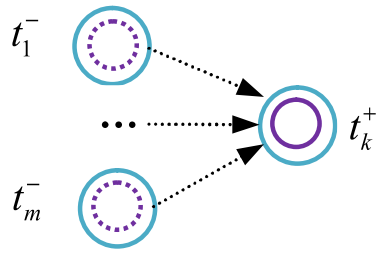

Figure 5.4: Simplified n-gram role language model.

As shown in Figure 5.4, all modifier-terms (e.g., t_1^- and t_m^-) are dependent on the term t_k^+ that represents the kernel-object. Even for $n > 2$, the n-gram is at most a bigram level modeling. For the vocabulary $\mathbb{V}$ with $|\mathbb{V}|$ terms the intent role of which are annotated, the number of parameters to be estimated for the simplified n-gram role language model is decreased to $|\mathbb{V}|^2 + 2|\mathbb{V}|$.

Essentially, the core of SNGR corresponds to a model lexicon θ, which consists of two parts: The 1-gram part consists of $2|\mathbb{V}|$ independent parameters of the form $p(t_i^-)$ and $p(t_i^+)$, where $p(t_i^-)$ denotes the probability of term t_i being a modifier and $p(t_i^+)$ denotes the probability of term t_i being a kernel-object. The quasi-bigram part consists of $|\mathbb{V}|^2$

independent parameters of the form $p(t_j^-|t_i^+)$, which denotes the probability that term t_j being a modifier on condition that term t_i being a kernel-object. The process of generating a role-explicit query under the SNGR model can be described as: The kernel-object t_i is first generated with a probability of $p(t_i^+)$. Concerning the kernel-object t_i^+, the model will then select a set of mutually independent modifiers $\{t_j^-\}$, each of which holds the probability of $p(t_j^+|t_i^+)$, finally a role-explicit query is formulated. Not surprisingly, the simplified n-gram role model thus can be viewed as an instance of the latent model Ψ that models how a role-explicit query is generated from a finite vocabulary of kernel-objects and modifiers.

Without loss of generality, for a Chinese query with m characters (the English query is similar), say, $q^+ = c_1c_2...c_m$, there are a total of 2^{m-1} possible segmentations, say, $\{A_j = t_1t_2...t_n|j = 1, ..., 2^{m-1}\}$, where A_j denotes one segmentation, $t_i = c_{o_i}c_{o_i+1}...c_{o_{i+1}-1}$, $1 = o_1 < o_2 < ... < o_{n+1} = m + 1$. Let $|A_j|$ denote the number of terms in A_j, for the segmentation A_j, there are $|A_j|$ possible annotations of intent roles: $\{A_j^k = t_1^-...t_{k-1}^-t_k^+t_{k+1}^-...t_n^-|k = 1, ..., |A_j|\}$, where A_j^k denotes one possible annotation corresponding to the segmentation A_j, i.e., a kernel-object plus a set of modifiers. Under the SNGR model that represented as its model lexicon θ, the generating probability of one possible annotation A_j^k can be given as:

$$p(A_j^k|\Psi) = p(A_j^k|\theta) = p(t_k^+|\theta) \prod_{i;i\neq k} p(t_i^-|t_k^+, \theta) \tag{5.7}$$

Let $\Omega = \{A_j^k|j = 1, ..., 2^{m-1}; k = 1, ..., |A_j|\}$ denote all the possible annotations with respect to a role-explicit query q^+, the task of intent role annotation can be accomplished by searching A^* in Ω, satisfying:

$$A^* = argmax_{A_j^k \in \Omega} p(A_j^k|\theta, q^+) \tag{5.8}$$

5.3.3 Obtaining the Model Lexicon

The system proposed in Chapter 4 enables us to automatically obtain a repository of sufficient role-explicit queries. With the availability of large amount of query logs, little cost is required to establish a corpus of sufficient role-explicit queries. Suppose that we have

obtained a corpus of sufficient role-explicit queries $\Phi = \{q_i^+ = w_1^- ... w_{k-1}^- w_k^+ w_{k+1}^- ... w_n^- | i = 1, 2, 3...\}$, we introduce how to learn the model lexicon θ in a statistical way. Corresponding to Φ, there exist a vocabulary $\mathbb{V}$ of terms annotated kernel-object and/or modifier. Using the maximum likelihood estimation method, we estimate the 1-gram parameters in the model lexicon θ as:

$$p(t_i^+) = p(t_i)\frac{c(t_i^+) + 1}{c(t_i) + 2} \tag{5.9}$$

$$p(t_i^-) = p(t_i)\frac{c(t_i^-) + 1}{c(t_i) + 2} \tag{5.10}$$

where $c(t_i)$ means the number of times term t_i occurs in corpus, $c(t_i^+)$ means the number of times term t_i occurs as a kernel-object, $c(t_i^-)$ means the number of times term t_i occurs as a modifier. $p(t_i)$ reflects the likelihood that term w_i occurs, it is estimated using the *Laplace Smoothing* technique as:

$$p(t_i) = \frac{2 + c(t_i)}{2|\mathbb{V}| + \sum_{t_j} c(t_j)} \tag{5.11}$$

To handle the terms that did not occur in the learned model lexicon, we include a universal symbol $< unk >$ for unknown terms. The quasi-bigram parameters are smoothed with a unigram probability using an interpolation technique named *Absolute Discounting Smoothing* [66], which is expressed as:

$$p(t_j^-|t_i^+) = \frac{max\{c(t_j^-, t_i^+) - DI, 0\}}{\sum_{t_j} c(t_j^-, t_i^+)} + (1 - \lambda)p(t_j^-) \tag{5.12}$$

where $c(t_j^-, t_i^+)$ means the number of co-occurred times of t_j and t_i in the query log on the condition that term t_j is a modifier and term t_i is a kernel-object. In Equation 5.12, the linear interpolation factor $1 - \lambda$ is calculated as:

$$1 - \lambda = \frac{DI}{\sum_{t_j} c(t_j^-, t_i^+)} N_{1+}(t_i) \tag{5.13}$$

$$DI = \frac{N_{one}}{N_{one} + 2N_{two}} \tag{5.14}$$

where $N_{1+}(t_i)$ means the number of unique terms that co-occurred with t_i, DI is a discounting constant, N_{one} and N_{two} are the total number of term pairs with exactly one and two counts.

Table 5.1 illustrates two parameters in model lexicon θ learned in experiments (we use the logarithm of a probability to avoid small numbers in calculation).

t_i= 哈利波特(Harry Potter)	
$logp(t_i^-)$= -5.1715 ,$logp(t_i^+)$ = -3.6915	
t_j =壁纸(wallpaper)	$logp(t_j^-\|t_i^+)$ =-3.1870
t_j =游戏(game)	$logp(t_j^-\|t_i^+)$ =-1.4665
t_i =个人所得税(personal income tax)	
$logp(t_i^-)$ =-5.6481 , $logp(t_i^+)$ =-3.8772	
t_j =改革(reform)	$logp(t_j^-\|t_i^+)$ =-2.3890
t_j =美国(American)	$logp(t_j^-\|t_i^+)$ =-3.0543

Table 5.1: Example parameters in the model lexicon θ (10 based log).

As shown in Table 5.1, 哈利波特(Harry Potter) is often used as a name entity that may refer to a famous film or a popular game, therefor it has a larger probability to be submitted as a kernel-object. 壁纸(wallpaper) and 游戏(game) are two modifiers, which are commonly used to restrict 哈利波特(Harry Potter) into a particular aspect. Under the model lexicon θ, we can deduce that: role-explicit query 哈利波特游戏(Harry Potter game) is likely derived from a kernel-object 哈利波特(Harry Potter) plus a modifier 游戏(game) with a probability of -5.158 (10 based log) larger than any other possible annotations, e.g., 哈利(Harry) as the kernel-object with 波特(Potter) and 游戏(game) as the modifiers. Analogously, the same situation holds true for 个人所得税(personal income tax) oriented queries.

For a role-explicit query q^+, we segment it into a number of terms in all possible ways. For each possible segmentation A_j, we annotate it with intent roles in all possible ways. For each possible annotation A_j^k, we calculate its generating probability using the learned model lexicon θ. Finally the annotation with the highest probability is selected as the optimal result for IRA. The calculation seems infeasible as the number of possible annotations grows exponentially with query length. Fortunately, most role-explicit queries are short [86], the number of terms per English query is small on average (2.35) (6.41 characters for Chinese queries through our analysis in Section 3.2.3. The calculation thus

can be performed efficiently indeed.

5.4 Role-explicit Query Identification

For role-explicit query identification, we treat it as a binary classification problem. The adopted features are listed as follows:

(1) *Usage of SNGR*: The SNGR model quantifies the generative probability of a role-explicit query. For an arbitrary query q, if we assume it is role-explicit, under the SNGR model we can obtain a probability, which reflects the likelihood of q being a role-explicit query. Intuitively, for a positive instance that q is really role-explicit, the obtained probability will be reasonably high; otherwise, a negative instance that q is role-implicit will lead to a zero probability or an extreme low probability. To capture this intuition, we consider the generative probability quantified by the SNGR model as a training feature.

(2) *Query length*: Keyword queries simply consist of several terms and show no grammatical structure. While queries constructed as sentences or questions contain more terms and show obvious syntactical structure. The length of a query thus reflects its complexity to some extent. For a query that includes named entities, especially places and organizations, it commonly has a larger character-length. However, these named entities act as one semantic unit indeed. So the term-length is more reasonable than the raw character-length to reflect a query's complexity. Role-explicit queries are mostly short queries and show no obvious grammatical structure. To differentiate from the verbose queries that rarely occur as role-explicit queries, the term-length of a query is employed as a training feature.

(3) *Named entity*: Kernel-objects in role-explicit queries mainly consist of named entities, nouns or noun phrases. Therefore, including named entities or not is a positive feature for identifying role-explicit queries. Instead of performing sophisticated named entity recognition techniques, we rely on a Wikipedia-based approach. *Wikipedia*, which is known as the largest online encyclopedia, contains millions of entries. Most of the entries are typical named entities or keyword phrases throughout a large number of domains. Many named entity related works employ Wikipedia as their knowledge resource [44, 78]. In our study, for a specific query, we firstly segment it into a sequence of terms [104]. A query is assumed to contain a named entity if at least one of its composing terms appears

as an entry of Wikipedia.

(4) *Interrogative terms*: This feature is inspired by the observation that many question queries tend to contain interrogative terms, e.g., 树有多少种类(How many kinds of trees are there), the meaning of which can't be well expressed with kernel-object and modifier. Moreover, the interrogative terms are rarely observed in role-explicit queries. For this reason, we collect a set of frequent interrogative terms (Table 4.3 in Section 4.3.2) as feature terms when training our classifiers.

(5) *User distribution*: The definition of kernel-object and modifier derives from the different utility of terms performed by users in expressing their intents. Despite the simplicity, they effectively depict the concise structure of role-explicit queries. For a specific query, a large population of users submitting the same query implies a consistency with the utility of composing terms when clarifying an information need. While the number of users who formulate the same complex query, e.g., question query, tends to drop quickly. For this reason, we count the number of unique users who submit the same query as a training feature.

Compared with the features (1), (2), (3) and (4) that stem from the inner characteristics of an individual query, the feature (5) can be viewed as an external feature aggregated from different users. When performing role-explicit query identification with the machine learning classifiers, the SNGR model learned in Section 5.5.1 is used to calculate the generating probability (gp), which is denoted as a float value. The raw values of the query length (ql) and user distribution (ud) are directly used. The existence/absence of named entity (ne) and interrogative term (iw) are denoted as a 1/0 value. The target class attribute of role-explicit is denoted as yes/no. A number of machine classifiers are investigated in Section 5.5.2.

5.5 Performance Evaluation

5.5.1 Experiment for Intent Role Annotation

Metric

Similar to the evaluation of role-explicit query extraction in Section 4.5.2, we evaluate the performance of intent role annotation using term-level and query-level metrics respectively. The term-level metric detailed in Section 4.5.2 is directly adopted. Different from the query-level metric in Section 4.5.2, this time the top-n annotations of a candidate algorithm are considered (like the metric for NERQ [43]). In particular, the result of an algorithm is viewed as correct if at least one of the top-n results is the same as the standard annotation, i.e., the same set of terms, as well as the same intent role of each term. Based on the query-level metric in Section 4.5.2, let Θ be the standard annotation of role-explicit queries, $N(\Theta^{'})$ denote that we take the top-N annotations of each test instance into account, the *Top N Accuracy* ($Q_{pre}@N$) is defined as the fraction of top-n correctly annotated queries over the standard annotations:

$$Q_{pre}@N = \frac{\Theta \cap N(\Theta^{'})}{\Theta} \tag{5.15}$$

Data and Baseline

For evaluate intent role annotation, we split SogouQ into two parts: the former 24-day part is used to learn the model lexicon of SNGR. The latter 6-day part is used to construct the testing collection. We randomly extract queries from the 6-day part and manually construct a test collection consisting of 500 role-explicit queries. The intent roles of each query are manually annotated.

To our knowledge, there is no prior work conducted for IRA as defined in this paper, in order to evaluate the overall performance of the proposed framework, we adopt the traditional language model based approach for named entity identification as the baseline [41]. Given a sequence of terms $\{t_1...t_m\}$, named entities can be identified by searching the *Viterbi path* [53]:

$$< \hat{c_1}...\hat{c_m} >= argmax_{c_1...c_m} p(c_1, t_1, ..., c_m, t_m) \tag{5.16}$$

where $c_1 ... c_m$ are named entity classes. Indeed the model defined by Equation 5.16 is equivalent with the one depicted by Equation 5.5. The difference is that the model represented by Equation 5.16 works on English terms, while Equation 5.5 considers all possible segmentations of a Chinese query. They both determine the optimal annotation through sequential maximum likelihood estimation. Therefore, we employ the 2-gram role language model and 3-gram role language model as baselines, denoted as *2-NGR* and *3-NGR* respectively. 2 – NGR and 3-NGR represent a straightforward utilization of the language modeling approach.

Experimental Results

Table 5.2 summarizes the overall performance through the term-level metric.

Approach	T_{pre}	T_{rec}	$T_{f-score}$
2-NGR	31.38%	17.60%	22.55%
3-NGR	40.75%	24.81%	30.84%
SNGR	78.06%	74.64%	76.31%

Table 5.2: Term level performances.

Table 5.3 summarizes the overall performance through the Top N Accuracy metric.

Approach	Q_{pre}@1	Q_{pre}@2	Q_{pre}@3
2-NGR	30.20%	46.00%	52.20%
3-NGR	36.20%	54.40%	58.20%
SNGR	73.20%	82.20%	85.40%

Table 5.3: Query level performances.

The term level metric reveals the accuracy that the intent role of a term inside a query can be predicted. The query level metric becomes more rigid in judging how well the predicted annotation of a query matches the standard annotation. As shown by Tables 5.2 and 5.3, 3-NGR performs slightly better that 2 – NGR across each metrics. It implies that 3-NGR is more robust than 2-NGR when encapsulating term attributes. The SNGR model outperforms all the baseline methods. The reasons can be summarized: SNGR emphasizes the dependency between modifier and kernel-object inside a query, and weakens the association among modifiers. Put another way, SNGR restricts the rigid bigram to term pairs consisting of kernel-object and modifier. 2 – NGR and 3-NGR treat kernel-object and modifier with an equal statistical way. As queries are often short, SNGR

is more robust to model the user queries with intent roles.

5.5.2 Experiment for Role-explicit Query Identification

For the evaluation of role-explicit query identification, we construct a hand-annotated corpus consisting of 1000 user queries, which are randomly extracted from the 6-day part. Using the features detailed in Section 5.4, each query is represented as a feature vector $< gp, ql, ne, iw, ud >$. To investigate the effectiveness of the proposed features, we use the *WEKA* toolkit [45] to build different classifiers including *Naïve Bayes*, *Decision Tree (J48)*, *Support Vector Machine (LibSVM)* and *AdaBoost (AdaBoostM1)*, and Naïve Bayes is selected as the baseline classifier. These supervised learners are well suited to classification tasks. For each classifier, we utilize the k-fold cross validation strategy, which can prevent the overfitting problem to some extent. We set k=10 in our experiment. The performances are evaluated in terms of *weighted average precision* (wa_p), *weighted average recall* (wa_r) and *weighted average f-measure* (wa_f) used in *WEKA*.

Table 5.4 summarizes the overall performances of different machine learning classifiers.

Classifier	wa_p	wa_r	wa_f
Naïve Bayes	0.823	0.498	0.528
Decision Tree	0.906	0.907	0.907
SVM	0.894	0.896	0.887
AdaBoost	0.896	0.898	0.897

Table 5.4: Overall performance of each classifier.

As shown in Table 5.4, the Naïve Bayes classifier performs the worst. The strong independence assumption underlying this classifier is that the presence or absence of a particular feature is unrelated to any other features. Therefore, a possible reason for its poor performance could be the existence of dependencies among the features. The SVM classifier separates the target classes with the maximal margin in a higher dimensional space. The AdaBoost classifier is an algorithm for constructing a strong classifier through combining a set of weak classifiers. Table 5.4 shows that they generate moderate results. Different from other classifiers that jointly consider a set of features to discriminate targets classes in a single decision step, the Decision Tree classifier performs classification in a multistage step based on a tree-like graph. Table 5.4 shows that the Decision Tree classifier achieves the best result. Moreover, the acceptable results of SVM, AdaBoost and Decision

Tree justify the way of identifying role-explicit queries in a supervised manner.

To further investigate the correlation between different feature combinations and classification performance, we enumerate all feature combinations and test their effectiveness. As the Decision Tree classifier performs the best, we utilize it for each feature combination. Table 5.5 shows all the possible feature combinations. Each row consists of the combinations with the same number of features. The combination in each row that generates the best result is underlined, and the best result is listed on the left 3 columns.

wa_p	wa_r	wa_f	Feature combination
0.906	0.907	0.907	<ud,ql,ne,iw>, ⋯, <u><gp,ql,ne,iw></u>
0.901	0.903	0.902	<ud,ne,iw>, ..., <u><gp,ql,iw></u>
0.893	0.896	0.894	<ud,iw>, <u><gp, ql></u>, ⋯, <gp,ud>
0.881	0.879	0.880	<gp>, <u><ql></u>, ⋯, <iw>

Table 5.5: Performances with respect to different feature combinations.

From Table 5.5, we can observe that:

(1) The results of one-feature based classification show that the query length at a term granularity is the most discriminating feature;

(2) Feature combinations with more features generate better results than feature combinations consisting of fewer features, which demonstrates that different features hold different discriminative abilities;

(3) the same result of $< gp, ql, ne, iw, ud >$ and $< gp, ql, ne, iw >$ indicates that feature ud shows no positive effect when combined with other four features.

Chapter 6

An Application in Subtopic Mining

6.1 Introduction

In Chapter 3, the idea of intent role oriented query parsing is proposed and preliminarily discussed. Furthermore, in Chapters 4 and 5, we have investigated the fundamental issues with respect to intent role oriented query parsing, i.e., role-explicit query extraction (Chapter 4), intent role annotation (Chapter 5) and role-explicit query identification (Chapter 5). The proposed approaches and models for these issues pave the way for further exploring the potential value of intent role oriented query parsing.

In this chapter, we focus on the matter of how well the idea of intent role oriented query parsing can work in reality. In particular, based on the idea of intent role oriented quer parsing, we study the problem of subtopic mining. The strategy of subtopic mining via modifier graph clustering is proposed, the core of which is the modifier graph derived from intent role oriented parsing of subtopic strings. Based on the standard test collections and data released by NTCIR-10 workshop, a series of experiments are conducted. As demonstrated by the experimental results, the proposed strategy of subtopic mining via modifier graph clustering outperforms the two baseline methods and shows a stable performance against the topics that have sparse query-level knowledge in a query log.

The rest of this chapter is organized as: Section 6.2 describes the motivation for subtopic mining. In Section 6.3, we review the state-of-the-art algorithms proposed for

query intent reference, which are also applicable to subtopic mining. Moreover, we also give a brief introduction of graph clustering, which plays an important role in our modifier graph clustering. Section 6.4 formally defines the problem of subtopic mining. Section 6.5 introduces the concept of modifier graph and details how to construct a modifier graph. Section 6.6 details the strategy of subtopic mining via modifier graph clustering. Section 6.7 describes experimental results and discussions.

6.2 Motivation

Recent studies show that the vast majority of users are submitting short queries with little or no context, which are often ambiguous and/or underspecified [1, 25, 72]. As discussed in Section 1.1, the query *Harry Potter* could refer to a book or a movie. In the category of *movie*, a user may be interested in the main character or the reviews.

To accommodate different search intents, a number of studies [2, 76] proposed the strategy of web page clustering. The idea is that: For a query, the retrieved web pages satisfying the same intent or information need should be grouped into a subset, by which the search result is diversified. This strategy has a drawback that the relative importance of an intent can't be obtained by simply clustering the retrieved web pages [73].

Compared with web page clustering, the technique of *search result diversification* has attracted significant attention. It features a trade-off between relevance (ranking more relevant web pages in higher positions) and diversity (satisfying users with different information needs) [1, 3, 21, 24, 72, 74, 82]. In particular, the entire problem of search result diversification can be decomposed into two joint subproblems: *subtopic mining* and *document ranking* [79, 89]. Subtopic mining aims at mining the possible subtopics behind a query, where a subtopic equivalently represents a unique intent or an information need. By combining the obtained subtopics at the first step of subtopic mining, document ranking aims at providing a diversified result to satisfy a large population of users. Moreover, there are workshop tasks (INTENT task of NTCIR-9 & NTCIR-10) corresponding to the two subproblems. Standard test collections and data are released, which facilitates the research progress in this field.

In this thesis, we focus on the subproblem of subtopic mining, and integrating the

results of subtopic mining for the subsequent document ranking is planed as the future work.

6.3 Related Work

6.3.1 State-of-the-Art Algorithms of Subtopic Mining

Compared with the problem of query intent inference (Section 2.5.3) that has a wide coverage, subtopic mining is more specific and has standard formats with respect to input and output. While many algorithms proposed for query intent reference (e.g., query suggestion, query clustering) can be adapted to the problem of subtopic mining, such as [46, 73, 77].

In particular, As studied by Sadikov et al. [77] and Radlinski et al. [73], the web page click and session co-occurrence information are positive indicators of queries' relevance. Sadikov et al. [77] built the Markov graph to model the relationships among the refinements of a query. By representing a query as the distribution over the clicked web pages, the refinements of the input query are clustered to represent different intents. In the study by Radlinski et al. [73], two refinements of a query are connected if the corresponding random walk similarity over the bipartite query-document click graph exceeds a pre-defined threshold. Based on the obtained graph of refinements, the random walk similarities are used to find intent clusters that consist of refinements. The key feature adopted by Hu et al. (2012) was that: many users add one or more additional keywords to expand the query to clarify their search intents. In their study, a subtopic was represented by keywords and URLs.

Most of the above methods have treated a whole query as the minimum analysis unit. For previously unseen queries or queries that have sparse relevant user behaviors in the query log, their performances are heavily impacted. In this thesis, we look into queries at a fine-grained term level, as well as the subtopic strings that are not included in a query log. Based on the previously defined intent roles, we propose the concepts of co-kernel-object elements and modifier graph. When the edges of the modifier graph have been quantified using the TKQL in a query log, the partition of a modifier graph helps us to derive the underlying subtopics of a given test topic. We believe that our modifier-graph

based approach will be useful for robust subtopic mining, especially for tail queries.

6.3.2 Graph Clustering

What is Graph Clustering

A *graph* is a representation of a set of nodes (or vertices) and a set of edges, where some pairs of nodes are connected by edges. Taking into consideration the edge structure of the graph, *graph clustering* is defined as a task of grouping the nodes into clusters in such a way that nodes within each cluster are densely connected and sparsely connected between clusters as well. A cluster is also called a *community* in some literatures like [37, 40]. As a hot research topic, graph clustering is widely explored from different dimensions. For global clustering, each node of the input graph is assigned a cluster. Different types of clustering algorithms are proposed, e.g., divisive clustering [15, 36] and agglomerative clustering [32]. In contrast, local clustering performs cluster assignments for a certain subset of nodes of a graph, such as [83].

Moreover, for identifying a good clustering, a number of measures and approaches have been proposed [30, 65]. In particular, for a weighted graph, the *modularity* [65] over a particular clustering of k clusters $C_1, ..., C_k$ is defined as:

$$M(C_1, ..., C_k) = \sum_{i=1}^{k} \varepsilon_{i,i} - \sum_{i,j \in \{1,...,k\}; i \neq j} \varepsilon_{i,j} \tag{6.1}$$

where $\varepsilon_{i,j} = \sum_{\{v_m, v_n\} \in E; v_m \in C_i, v_n \in C_j} w(v_m, v_n)$ with each edge included at most once in the computation. Modularity has been widely used to evaluate the clusterings obtained by different algorithms, but also as an objective function to optimize [26, 32], i.e., the higher modularity an algorithms achieves, the better this algorithm is.

As graph clustering is a quite popular and broad topic, we are not going to provide a state-of-the-art survey, but rather than an explanation of the methods related to our work. For an exhaustive survey of the methodologies in this field, we refer the reader to the literatures [37, 48, 84].

Graph Clustering in Our Study

As discussed by Schaeffer [84], clustering is likely to be application specific, as well as graph clustering. In our work, graph clustering is performed in context of a weighted graph (modifier graph in Section 6.5). The key points that guide our choice are detailed in Section 6.6.2. Taking into consideration the requirements of parameter-free and efficiency, we are interested in two methods of *Louvain* and *LinLog*.

Blondel et al. [10] proposed the Louvain method for community detection. The Louvain method performs community detection by optimizing modularity (Equation 6.1) in a heuristic manner. As shown by Blondel et al. [10], the Louvain method can find high modularity partitions of an input graph and outperforms all other known community detection methods in terms of computation time. Moreover, this method can easily be adapted to weighted and directed graph, and it is also implemented in several network analysis softwares, e.g., *NetworkX*[1] and *Gephi*[2].

Noack [67] proposed the LinLog algorithm for uncovering the cluster structure of an input graph. Two energy models *node-repulsion* and *edge-repulsion* are introduced, in particular, node-repulsion energy model bases on repulsion between nodes and edge-repulsion energy model bases on repulsion between edges. As shown by Noack [67], these two energy models are closely related to the widely used quality criteria for graph clustering, e.g., the density of the cut [56, 62], modularity, etc.

6.4 Problem Formulation

We begin by providing the basic definitions and then formulate the subtopic mining problem. The "Subtopic Mining" task of NTCIR[3] and the "Diversity" task of TREC Web Track[4] are the core prototypes on which our formulation builds.

Definition 18 (Topic) *A topic (denoted as $\ddot{T}$) refers to a test instance for evaluating a subtopic mining system.*

[1] http://networkx.github.io/
[2] http://gephi.org/
[3] http://research.microsoft.com/en-us/projects/intent/
[4] http://plg.uwaterloo.ca/ trecweb/

Definition 19 (Subtopic) *A subtopic (denoted as $\ddot{t}$) refers to a possible information need or an intent underlying a topic $\ddot{T}$.*

A subtopic could be an interpretation of an ambiguous topic or an aspect of an underspecified topic. When performing subtopic mining, it is assumed that the possible subtopics are mutually independent.

Definition 20 (Subtopic String) *A subtopic string (denotes as tStr) is viewed as an expression of a subtopic.*

As an illustration, Figure 6.1 shows a dry-run topic released during the *INTENT* task of NTCIR-9[5].

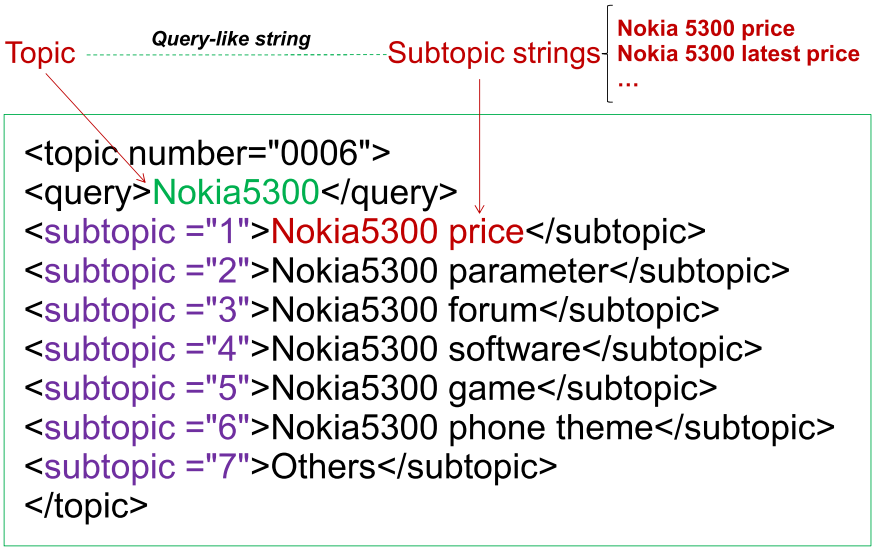

Figure 6.1: A dry-run topic.

As shown in Figure 6.1, there are 7 subtopics labeled by human assessors. For the subtopic *Nokia 5300 price* (a representative subtopic string), *Nokia 5300 price* and *Nokia 5300 latest price* are different subtopic strings that express the same subtopic. Put another way, multiple subtopic strings may express the same subtopic. Moreover, a topic and a subtopic string can either be a real query or a query-like string derived from other resources, e.g., query suggestions provided by a web search engine or segments extracted from a document collection.

Formally, for a given topic $\ddot{T}$, suppose there are k possible subtopics $\{\ddot{t}_1, ..., \ddot{t}_k\}$. Let $\{tStr_1, ..., tStr_n\}$ be a set of n subtopic strings with respect to $\{\ddot{t}_1, ..., \ddot{t}_k\}$. The problem

[5]http://www.thuir.org/intent/ntcir9/

of subtopic mining is formulated as finding a ranked list of subtopic strings L that ranks subtopic strings representing more popular subtopics at higher positions and includes as many subtopics as possible. The quality of L depends on its consistency with the ideal ranked list L^* (e.g., the ranked list created by human assessors).

6.5 Modifier Graph

Definition 21 (Co-kernel-object Elements) *Given a particular kernel-object ko, co-kernel-object elements refer to a set of role-explicit subtopic strings that share the same kernel-object ko, denoted as* $CoKO(ko) = \{tStr\}$.

As an illustration, Table 6.1 illustrates three role-explicit subtopic strings, the kernel-object and modifier of each subtopic string are annotated at the same time. Moreover, whether each subtopic string is included in the query log SogouQ is marked at the 3rd column. According to the Definition 21, the three role-explicit subtopic strings in Table 6.1 are co-kernel-object elements, where 哈利波特(Harry Potter) is the shared kernel-object. As discussed in Section 3.4, the co-kernel-object elements can be viewed as different expressions of kernel-object oriented subtopics, the differences among which essentially are revealed by the modifiers, such as 游戏(game) and 小说(fiction).

Subtopic String	Intent Role Annotation	In SogouQ?
哈利波特游戏(Harry Potter game)	*ko*: 哈利波特(Harry Potter) *mo*: 游戏(game)	Yes
哈利波特小说(Harry Potter fiction)	*ko*: 哈利波特(Harry Potter) *mo*: 小说(fiction)	Yes
哈利波特阅读(Harry Potter reading)	*ko*: 哈利波特(Harry Potter) *mo*: 阅读(reading)	Not

Table 6.1: Role-explicit subtopic strings.

Because subtopic strings essentially are query-like strings, given a group of subtopic strings, the query-level knowledge in query log (QKQL, Section 3.2.2) can be derived from a query log if these subtopic strings are included in the query log. Analogously, we can derive the term-level knowledge in query log (TKQL, Section 3.2.2) for kernel-object and modifier once we treat them as terms. For instance, based on the query log SogouQ, Table 6.2 shows the TKQL with respect to each pair of modifiers in Table

6.1, i.e., the statistics of co-query, term-level co-session and term-level co-click (given as *CoQuery:TCoSession:TCoClick*), where the self-loop is not considered.

	游戏(game)	小说(fiction)	阅读(reading)
游戏(game)	×	7:78:20	1:13:1
小说(fiction)	7:78:20	×	700:993:3637
阅读(reading)	1:13:1	700:993:3637	×

Table 6.2: Term-level knowledge in query log (TKQL) derived from SogouQ.

As shown in Table 6.2, the TKQL hidden in a query log differs a lot with respect to different pairs of modifiers. For example, for 小说(fiction) and 游戏(game), the corresponding TKQL is *7:78:20*, while that for 小说(fiction) and 阅读(reading) is *700:993:3637*. In the following sections, we will detail how to make use of the TKQL hidden in the query log for subtopic mining.

Definition 22 (Modifier Graph) *Modifier graph $G_{ko} = \{V, E\}$ is an undirected graph derived from a set of co-kernel-object elements $CoKO(ko)$, where:*

(i) The node set V consists of modifiers derived from $G_{ko} = \{V, E\}$, i.e., $V = \{mo|tStr \in CoKO(ko), mo \succ tStr\}$, where $mo \succ tStr$ denotes that mo is a composing modifier of subtopic string $tStr$.

(ii) The edge set E denotes the set of edges, i.e., $E = \{e = (mo_i, mo_j)|mo_i \in V, mo_j \in V\}$.

Assuming that $CoKO(ko = 哈利波特(HarryPotter))$ merely consists of the three subtopic strings shown in Table 6.1 (other subtopic strings that share the same kernel-object indeed exist), Figure 6.2 shows the corresponding modifier graph. Moreover, the edges are weighted based on the corresponding TKQL of each pair of modifiers (Table 6.2) with Equation 6.2.

$$\begin{aligned}\delta_{TKQL}(mo_i, mo_j) = &\lambda_1 * \frac{CoQuery(mo_i, mo_j)}{max_V CoQuery(mo_m, mo_n) + 1} + \\ &\lambda_2 * \frac{TCoSession(mo_i, mo_j)}{max_V TCoSession(mo_m, mo_n) + 1} + \\ &\lambda_3 * \frac{TCoClick(mo_i, mo_j)}{max_V TCoClick(mo_m, mo_n) + 1}\end{aligned} \tag{6.2}$$

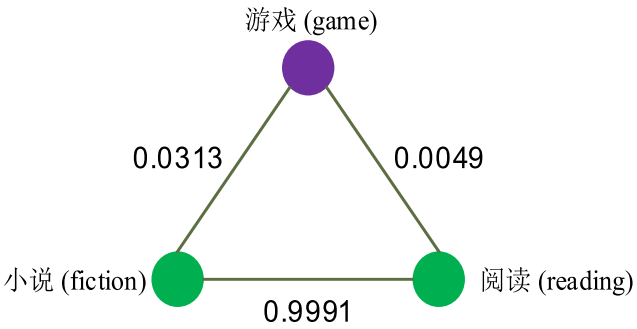

Figure 6.2: An example modifier graph.

Based on the edge weights shown in Figure 6.2, 小说(fiction) and 阅读(reading) are strongly interacted (an edge weight of 0.9991). 游戏(game) and 小说(fiction) (an edge weight of 0.0313)，游戏(game) and 阅读(reading) (an edge weight of 0.0049) are weakly interacted.

The prior studies [38, 46, 73, 77, 98] have shown that: the QKQL, i.e., web page co-click and session co-occurrence, are good indicators of the topical relevance of queries. Put another way, for two queries, the larger the count of co-session and/or the count of co-click are, the more relevant the two queries are. Because the TKQL derives from QKQL, it is reasonable to believe that: For two terms, the larger the count of co-query and/or the count of term-level co-session and/or the count of term-level co-click are, the more similar subtopics the two term express. In Equation 6.2, the statistics of co-query, term-level co-session and term-level co-click corresponding to a pair of modifiers are normalized by the maximum ones of pairs of modifiers within V respectively, i.e., $max_V CoQuery(mo_m, mo_n)$, $max_V TCoSession(mo_m, mo_n)$ and $max_V TCoClick(mo_m, mo_n)$. The weight essentially is a linear combination of the normalized values. Therefor, a higher value of $\delta_{TKQL}(mo_i, mo_j)$, the more similar of subtopics indicated by modifiers mo_i and mo_j.

If we perform graph clustering over the weighted modifier graph (Figure 6.2), 游戏*(game)* is likely to be grouped into a cluster, say, *Cluster-1*. 小说*(fiction)* and 阅读*(reading)* are likely to be grouped into another cluster, say, *Cluster-2*. According to the nature of TKQL, we can deduce that: 小说*(fiction)* and 阅读*(reading)* express a similar subtopic, 游戏*(game)* expresses a different subtopic. Due to the fact that the subtopics underlying the co-kernel-object elements largely depends on their composing modifiers, we can further deduce that 哈利波特阅读(Harry Potter reading) and 哈利波特小说(Harry

Potter fiction) express the same subtopic, while 哈利波特游戏(Harry Potter game) express a different subtopic. Not surprisingly, the partition of a modifier graph helps to uncover the prior unknown subtopics underlying a set of co-kernel-object elements. Based on this observation, we propose the strategy of modifier graph based subtopic mining.

6.6 Modifier Graph based Subtopic Mining

As observed in Section 6.5, for the modifier graph quantified with TKQL from a query log, the modifiers reflecting the same subtopic are strongly interacted, while the modifiers reflecting different subtopics are weakly associated. The partition of modifier graph helps us to uncover the latent subtopics underlying a group of subtopic strings. To perform modifier graph based subtopic mining, several key problems must be addressed:

(1) Modifier graph construction, which is the premise of modifier graph based subtopic mining;

(2) Modifier graph clustering, which aims at decomposing the modifier graph into a set of clusters. Each modifier cluster reasonably represents a possible subtopic;

(3) Generating the ranked list of subtopic strings, which is the target output. In the following subsections, we detail how we address these problems respectively.

6.6.1 Modifier Graph Construction

A quality modifier graph is characterized by the following two features: (1) The modifier graph must be derived from a set of sufficient subtopic strings, which ensures the coverage of possible subtopics; (2) The edges of the modifier graph must be reasonably quantified, which helps uncover the latent subtopics. Considering the above two requirements, we construct the modifier graph through the following three steps:

Step-1: Given a role-explicit topic T, we firstly identify its kernel-object ko using the algorithm introduced in Section 5.3.

Step-2: Given the identified kernel-object ko, we secondly obtain sufficient co-kernel-object elements $CoKO(ko)$.

Specifically, we first manage to obtain a set of candidate subtopic strings, represented as $\{ctStr\}$, where $ctStr$ denotes a candidate subtopic string. $\{ctStr\}$ is further used as the

base for generating the co-kernel-object elements. Here *recall* is the key point, it ensures the coverage of different subtopics. The resources we considered are:

- **Query Log**: We scan the whole query set, all the queries that include ko as a substring are selected as candidate subtopic strings.
- **Query Suggestion**: The recommended queries are often semantically related or are frequently performed by a large number of users. The query suggestions for topic T are directly selected as candidate subtopic strings.
- **Result Snippets**: We extract noun phrase (NP) and verb phrase (VP) segments from the result snippets returned by a search engine for topic T. The segments that including ko as a substring are selected as candidate subtopic strings.

According to the Definition 21 in Section 6.5, the co-kernel-object elements are all role-explicit ones that can be represented with kernel-object and modifiers. When deriving the co-kernel-object elements based on a set of candidate subtopic strings, we relax the requirement as: regardless of whether a candidate subtopic string is role-explicit or role-implicit, once it includes ko as a substring, we intuitively assume that it is a co-kernel-object element. The substring "ko" is directly regarded as its kernel-object, the remaining parts are directly segmented into modifiers (stop-words are discarded).

Step-3: Given the co-kernel-object elements $CoKO(ko)$, the node set V of the target modifier graph G_{ko} consists of modifiers of the elements in $CoKO(ko)$. We add an edge for each pair of distinct modifiers mo_i and mo_j iff $\delta_{TKQL}(mo_i, mo_j) > 0$ (Equation 6.2). Based on a pilot experiment, we let $\lambda_1 = \lambda_2 = \lambda_3 = \frac{1}{3}$ in this thesis.

6.6.2 Modifier Graph Clustering

The key points for modifier graph clustering are:

(1) For subtopic mining, the number of possible subtopics is not known as a priori and a manual setting can't guarantee a natural partition of the possible subtopics. The adopted clustering algorithm should solve this sensitive parameter problem in a reasonable way.

(2) Since subtopic mining systems are regarded as a subpart or an intermediate step for providing diversified search service, the clustering algorithm should be able to handle a certain amount of data under reasonable space and time conditions.

Suppose π denotes a graph clustering algorithm that does meet the above requirements and performs graph clustering efficiently without any input parameter (different algorithms are tested in Section 6.7). By performing modifier graph clustering, we aims at generating a set of subtopic string clusters, denoted as $SC = \{sc_k\}$. Each cluster of subtopic strings is used to represent a subtopic, i.e., $sc_k = \{tStr_m\} \rightarrow t_k$. The procedures are as follows:

Taking the modifier graph G_{ko} as the input of π, a group of modifier clusters are generated as output. We denote them as $MC = \{mc_k\}$, where $mc_k = \{mo_i\}$ represents a cluster of modifiers. Specifically, corresponding to each modifier cluster mc_k, we generate a subtopic string cluster sc_k, namely the cluster numbers are equal, i.e., $|MC| = |SC|$. For each modifier $mo_i \in mc_k$, we select the subtopic string $tStr_m$ that meets $mo_i \succ tStr_m$, and add $tStr_m$ into the corresponding subtopic string cluster sc_k. In particular, when allocating a subtopic string that its composing modifiers occur in different modifier clusters, we obey the following priority rules:

- Add it into the subtopic string cluster, of which the corresponding modifier cluster encloses more composing modifiers. For example, suppose three modifiers in $tStr_m$, namely, $mo_i \succ tStr_m$, $mo_j \succ tStr_m$ and $mo_r \succ tStr_m$. The three modifiers occur in two differ modifier clusters, e.g., $mo_i \in mc_k$, $mo_j \in mc_s$ and $mo_r \in mc_s$. Then $tStr_m$ is added into sc_s.

- If two modifier clusters enclose an equal number of modifiers, add it into the subtopic string cluster, of which the corresponding modifier cluster encloses the most frequent modifier (or the count sum of the enclosed modifiers). For example, suppose two modifiers in $tStr_m$, namely, $mo_i \succ tStr_m$, $mo_j \succ tStr_m$. The two modifiers occur in two different modifier clusters, e.g., $mo_i \in mc_k$, $mo_j \in mc_s$. Then $tStr_m$ is added into sc_s iff $|mo_j| > |mo_i|$, and vice versa. Here $|mo|$ represents the count of modifier mo in the node set.

The idea underlying the above rules are that: Larger number of modifiers means a

larger expressive power. A larger count of modifier means a larger expressive power that a rare modifier.

6.6.3 Generating the target ranked list of subtopic strings

We assume that there is a probability distribution for modifiers, say $p(mo)$, which is sampled repeatedly by users to form mutually-independent modifiers to clarify kernel-object oriented subtopics.

Definition 23 (Expression Power) *Expression power of a subtopic string is given as the generating probability of its composing modifiers, which measures its effectiveness of describing the subtopic represented by the subtopic string cluster it belongs to.*

In our study, the expression power (EP) of a subtopic string $tStr$ is given as:

$$EP(tStr) = \prod_{mo_i \succ tStr} p(mo_i) \tag{6.3}$$

As shown in Equation 6.3, expression power essentially is formulated as a unigram language model with a *gram* being a modifier. The value of expression power lies in that: we rely on the expression power to select the representative subtopic strings from a subtopic string cluster when generating the ranked list. For the required probability of $p(mo)$ in Equation 6.3, it is estimated using the Laplace Smoothing technique as:

$$p(mo_i) = \frac{|mo_i| + 1}{\sum_{\{mo_j \in V\}} |mo_j| + |V|^+} \tag{6.4}$$

where $|V|^+$ denotes the number of distinct modifiers in the node set V.

Definition 24 (Subtopic Popularity) *Subtopic popularity reflects the likelihood of a particular subtopic.*

In out study, the subtopic popularity is formulated as:

$$SP(t_k) = SP(sc_k) = \frac{|sc_k|}{\sum_{sc_s \in SC} |sc_s|} \tag{6.5}$$

where t_k corresponds to the subtopic represented by cluster sc_k, $|sc_k| = \sum_{tStr_m \in sc_k} |tStr_m|$ and $|tStr_m|$ denotes the count of $tStr_m$ in $CoKO(ko)$.

When generating the target ranked list L, the gain value of putting a subtopic string $tStr_r$ at the $r-th$ position is given as:

$$G(r) = \frac{SP(sc_k)}{log(r+1)} \tag{6.6}$$

where $tStr_r \in sc_k$, namely the gain value builds upon the subtopic popularity of the subtopic string cluster that $tStr_k$ belongs to. Then, the discounted cumulative gain of the ranked list L with a cutoff value r is given as:

$$DCG(r) = \beta * \frac{N_{1+}(r)}{|SC|} + (1-\beta) * \sum_{j=1}^{r} G(j) \tag{6.7}$$

where $N_{1+}(r)$ denotes the number of distinct clusters that the top-r subtopic strings cover. If we view the cluster number $|SC|$ as the number of possible subtopics, $\frac{N_{1+}(r)}{|SC|}$ represents the subtopic diversity. $\sum_{j=1}^{r} G(j)$ aims at ranking subtopic strings that describe popular subtopics at higher positions. Based on a pilot experiment, we let $\beta = 0.5$.

Given a required size (e.g., 30 or 50) of the target ranked list, L can be obtained iteratively based on the Definitions 23, 24 and Equation 6.7. Algorithm 4 illustrates the detailed algorithm.

Algorithm 4 The algorithm for generating the ranked list

Input: The required size of the ranked list, η; The set of subtopic string clusters, $SC = \{sc_k\}$;

1: **for** each $sc_k \in SC$ **do**
2: Rank the subtopic strings $tStr_m \in sc_k$ in order of their decreasing expression power values $EP(tStr_m)$;
3: **end for**
4: Select the subtopic string cluster sc^* that meets: $sc^* = \arg\max_{sc_k \in SC} SP(sc_k)$;
5: Select the subtopic string $tStr^*$ that meets: $tStr^* = \arg\max_{tStr_m \in sc^*} EP(tStr_m)$;
6: Add $tStr*$ at the first position of the target ranked list L and remove it from sc^*;
7: Set $j = 2$
8: **while** $j \leq \eta$ *and* $|SC| > 0$ **do**
9: Select the subtopic string $tStr^*$ that meets: $tStr^* = \arg\max_{\{sc_n \in SC\}}(DCG(j) - DCG(j-1))$, where $DCG(j) - DCG(j-1)$ denotes the gain when adding a subtopic string sc_n at the $j-th$ position of L;
10: Add $tStr^*$ at the $j-th$ position of L and remove $tStr^*$ from the subtopic string cluster which it belongs to;
11: $j++$
12: **end while**
13: **return** L

As shown in Algorithm 4, for one cluster sc_k, the subtopic string with a larger expression power is preferred to be its representation. Moreover, for the set of subtopic string clusters SC, the subtopic strings in a cluster that scores a higher subtopic popularity have a larger probability to be ranked at a higher position.

6.7 Performance Evaluation

6.7.1 Data

To conduct a convincing and reproducible evaluation, we resort to the official topic set and resources released by NTCIR-10 in the Intent task [79]. The official resources include the publicly available Chinese query log SogouQ and a uniform set of query suggestions for each topic. What is more important, the ideal ranked list of subtopic strings for each topic is available. We refer the reader to the official website for further details. As for search snippets for extracting NP & VP segments, the search engine of Google is adopted and the top-100 search snippets are used.

As discussed in Section 3.3, the role-implicit queries are often question queries or verbose queries, which merely require one specific answer. To solve topics of this type, there are specific methods [5, 71] in the field of question answering. In this thesis, we focus on role-explicit topics that generally have multiple subtopics, which are hard to cope with during information retrieval. Hence, 6 role-implicit topics (Table 6.3) are excluded from the entire topic set. We denote the resultant test set of role-explicit topics as *Topic-Set-A*.

ID	Topic
0244	2008年春节是哪天(what day is the spring festival in the year 2008)
0245	央视主持人周涛简历(resume of CCTV host ZhouTao)
0249	什么是京都议定书(what is Kyoto Protocol)
0256	什么是RTF (what is RTF)
0270	陕西临潼农民发现秦始皇兵马俑 (the farmer in Lintung of ShanXi discovered the Terracot-ta Army)
0283	鉴宝节目主持人是谁(who is the host of the treasure evaluation program)

Table 6.3: Role-implicit topics.

In order to investigate to what extent we can rely on the query-level knowledge in query log to perform subtopic mining, we did an analysis based on SogouQ. In Figure 6.3, the x-axes represent the topic IDs in Topic-Set-A (starts from 200). The y-axes denote

the count per topic. In particular, the asterisk implies the count of co-session queries, the circle represents the count of candidate subtopic strings that occurred in SogouQ.

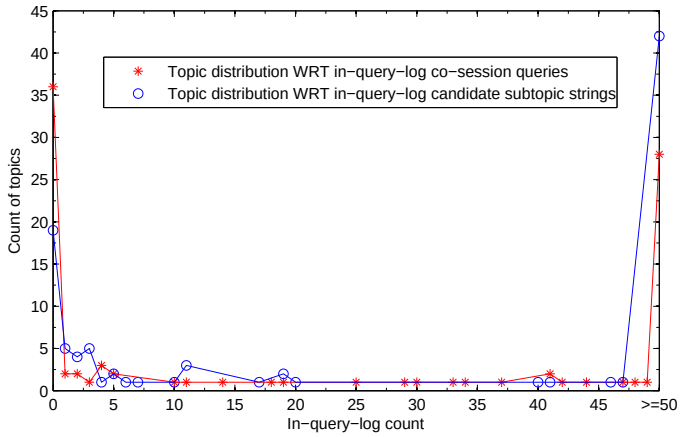

Figure 6.3: To what extent we can rely on SogouQ.

From Figure 6.3, we found that: 46 topics have co-session queries less than 10. 38 topics have candidate subtopic strings occurred in SogouQ less than 10. If we set the cutoff value as 10, it means that there are not enough subtopic strings to generate the ranked result list. For topics of this kind, the methods proposed by [73, 77] can't work well due to the sparseness problem.

By excluding the 38 topics, of which the count of candidate subtopic strings occurred in SogouQ is smaller than 10 (because the smallest cutoff value is 10 in our evaluation), we built another test set: *Topic-Set-B*, which is a subset of Topic-Set-A. Table 6.4 shows the information about the two test topic sets.

	Size	Property
Topic-Set-A	92	SogouQ includes little query-level knowledge for 38 topics
Topic-Set-B	54	Query-level knowledge in SogouQ can be used

Table 6.4: Two test topic sets.

6.7.2 Graph Clustering Algorithms

In our modifier graph based approach, the graph clustering algorithm is a key component for generating modifier clusters. To understand the impact that different graph clustering algorithms may have on our approach, the Louvain algorithm [10] and LinLog algorithm [67] discussed in Section 6.3.2 have been tested, both of which do not require a

predefined cluster number.

6.7.3 Evaluation Metric

We use the metric *D#-nDCG* [80] suggested by NTCIR workshop [79] as our evaluation metric, which is a linear combination of *I-rec* [106] and *D-nDCG* [17, 50]:

$$D\# - nDCG@l = \gamma * I - rec@l + (1 - \gamma) * D - nDCG@l \tag{6.8}$$

I-rec indicates the proportion of subtopics covered by the top subtopic strings and measures diversity. D-nDCG measures overall relevance across subtopics, which ranks subtopic strings that are highly relevant to major subtopics above those that are marginally relevant to minor subtopics. D#-nDCG balances subtopic diversity and subtopic relevance through the tuning parameter γ. In our study, we evaluate I-rec, D-nDCG, D#-nDCG with cutoff values of 10 and 20, and set $\gamma = 0.5$ as suggestion by [79].

6.7.4 Baseline Methods

For comparison purpose, two typical methods in the field of query intent inference are used as our baselines. They are:

Sadikov et al. [77] performed intent inference by clustering the query refinements. In particular, given a query and a set of its refinements, a Markov graph (nodes of which are the refinements) is built based on the document click and session co-occurrence information of these refinements. By performing multiple random walks on the obtained Markov graph, a refinement (a graph node) is represented by the vector of absorption distribution over a set of documents. Finally, the problem of query refinements clustering is reduced to a problem of Euclidean-vector clustering.

Deng et al. [31] proposed the Click Frequency Inverse Query Frequency model (CF-IQF) for query clustering. Under CF-IQF, each query is represented by a vector of clicked documents. Thus, the problem of query clustering is also reduced to a problem of Euclidean-vector clustering.

As did by Sadikov et al. [77], the complete-linkage clustering algorithm is used for the two baseline methods. For convenience, the methods proposed by Sadikov et al. [77]

and Deng et al. [31] are denoted as *RClustering* and *CFIQF* respectively. Corresponding to the two graph clustering algorithms Louvain and LinLog, we denote our modifier graph based approach as *MG-Lou* and *MG-Lin* respectively.

6.7.5 Experimental Results

Because the Complete-linkage clustering algorithm requires a predefined cluster number, thus different values (5, 10 and 15) were tested for the two baseline methods to obtain the best performance. Based on Topic-Set-B, Tables 6.5 and 6.6 show the results with cutoff values l=10 and 20.

	I-rec@10	D-nDCG@10	D#-nDCG@10
CF-IQF(5)	0.1782	0.1753	0.1768
CF-IQF(10)	0.3066	0.2688	0.2877
CF-IQF(15)	0.3093	0.2715	0.2904
RClustering(5)	0.0353	0.0333	0.0343
RClustering(10)	0.0595	0.0501	0.0548
RClustering(15)	0.0806	0.0672	0.0739
MG-Lou	0.4375	0.4375	0.4375
MG-Lin	**0.4650**	**0.4275**	**0.4462**

Table 6.5: Topic-Set-B based comparison with $l = 10$.

	I-rec@20	D-nDCG@20	D#-nDCG@20
CF-IQF(5)	0.1782	0.1168	0.1475
CF-IQF(10)	0.3066	0.1767	0.2417
CF-IQF(15)	0.3903	0.2328	0.3116
RClustering(5)	0.0353	0.0216	0.0285
RClustering(10)	0.0595	0.0327	0.0461
RClustering(15)	0.1024	0.0504	0.0764
MG-Lou	0.5579	0.3975	0.4777
MG-Lin	**0.5878**	**0.3921**	**0.4899**

Table 6.6: Topic-Set-B based comparison with $l = 20$.

As shown in Tables 6.5 and 6.6, the baseline methods exhibit different performance with respect to different cluster numbers, and both of the two baseline methods achieve a slightly better performance with a larger cluster number. However, the number of subtopics indeed varies from topic to topic. A predefined cluster number can't guarantee a natural subtopic mining. The two modifier graph based approaches (MG-Lou and MG-Lin) outperform the baseline methods a lot in terms of I-rec, D-nDCG and D#-nDCG.

The reasons are that:

(1) Different from the baseline method that relies on query-level knowledge in query log (QKQL), the modifier graph based approaches make use of term-level knowledge in query log (TKQL). Due to a fine-grained granularity, TKQL enables us to determine the subtopics of subtopic strings derived from different resources rather than limited queries included in a query log. As shown in Section 6.5, though the subtopic string 哈利波特阅读(Harry Potter reading) is not included in SogouQ, we can determine its underlying subtopic based on its composing modifier 阅读(reading).

(2) Leveraging on the two ad-hoc graph clustering algorithms, our modifier graph based approaches can solve the problem of a hard subtopic determination, i.e., a predefined cluster number.

Tables 6.7 and 6.8 show the results based on Topic-Set-A with cutoff values l=10 and 20.

	I-rec@10	D-nDCG@10	D#-nDCG@10
CF-IQF(5)	0.1130	0.1042	0.1086
CF-IQF(10)	0.1449	0.1352	0.1401
CF-IQF(15)	0.1506	0.1430	0.1468
RClustering(5)	0.0145	0.0160	0.0153
RClustering(10)	0.0391	0.0355	0.0373
RClustering(15)	0.0411	0.0373	0.0392
MG-Lou	**0.4111**	**0.4253**	**0.4182**
MG-Lin	0.4121	0.4117	0.4119

Table 6.7: Topic-Set-A based comparison with $l = 10$.

	I-rec@20	D-nDCG@20	D#-nDCG@20
CF-IQF(5)	0.1130	0.0714	0.0922
CF-IQF(10)	0.1449	0.0890	0.1169
CF-IQF(15)	0.2142	0.1242	0.1692
RClustering(5)	0.0145	0.0104	0.0125
RClustering(10)	0.0391	0.0230	0.0311
RClustering(15)	0.0543	0.0282	0.0413
MG-Lou	**0.5334**	**0.4024**	**0.4679**
MG-Lin	0.5309	0.3945	0.4627

Table 6.8: Topic-Set-A based comparison with $l = 20$.

Compared with Topic-Set-B, Topic-Set-A includes many topics that have rare co-session queries or queries occurred in SogouQ. For topics of this type, the methods that

rely on click information at a whole query level can't work effectively. As shown in Tables 6.7 and 6.8, the performance of the two baseline methods are greatly impacted due to the sparseness problem. In contrast, our modifier graph based approaches exhibit a stable performance. Leveraging on the term-level knowledge in query log, they can perform subtopic mining even with the topics with little query-level knowledge in query log.

Based on Topic-Set-A, Figure 6.4 illustrates the distribution of the cluster count per topic. The asterisk represents the official number of subtopics, which are classified by human assessors. The circle indicates the count of clusters when performing modifier graph clustering with the Louvain algorithm. The triangle indicates the count of clusters when performing modifier graph clustering with the LinLog algorithm.

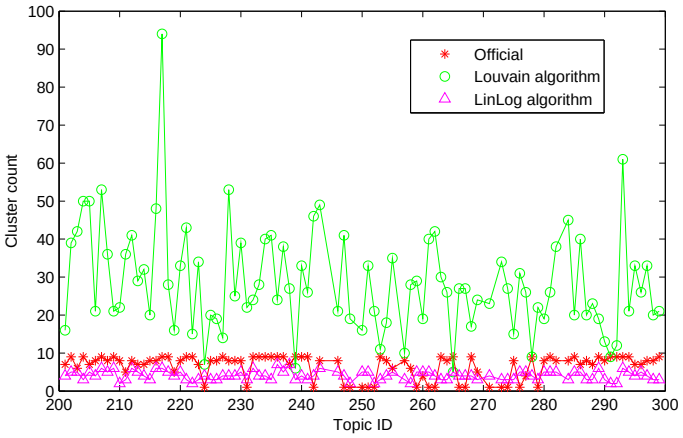

Figure 6.4: Cluster number distribution.

From Figure 6.4, we can observe that: The two graph clustering algorithms group the same modifier graph into different number of clusters due to different optimization criteria. The LinLog algorithm generally generates a small number of clusters compared with the Louvain algorithm. Unfortunately, both of the two algorithms commonly generate a different number of clusters compared with the official subtopic number. If we directly regard the cluster number as the subtopic number (as we did in Section 6.6.3), the approximated subtopic recall value would not be accurate enough. Therefore, in order to obtain a better subtopic mining result, a specific algorithm for clustering the modifier graph should be considered instead of using the existing algorithms.

Chapter 7

Conclusions and Future work

7.1 Conclusions

For better query representation and understanding, this thesis has proposed the strategy of intent role oriented query parsing, a novel scheme for performing query analysis.

In Chapter 3, we proposed our observation about the utilization of terms in the context of a query: The composing terms of a query play different role in manifesting the information need or intent carried by the query. Intuitively, we proposed two intent roles *kernel-object* and *modifier* to differentiate the terms within a query. Based on the preliminary analysis of intent role annotation, we find that: intent role oriented query parsing not only enables the query decomposition and understanding per-query, but also provides a way of mining the knowledge (or wisdom of crowds) hidden in massive user queries in a aggregative manner. In essence, the concept of modifier graph detailed in Chapter 6 is a good example of aggregative usage of massive annotated queries.

In Chapter 4, the proposed approaches for role-explicit query extraction mainly rely on the knowledge hidden in users' reformulating behaviors, such as the longest common substring of a pair of consecutive queries and the session-level contextual information. Their effectiveness in role-explicit query extraction in turn demonstrates the feasibility of parsing user queries based on the contextual information.

In Chapter 5, this thesis investigated the fundamental issues concerning intent role oriented query parsing. Instead of using the supervised methods, we resorted to the statistical approaches for intent role annotation. In particular, we proposed the simplified

n-gram role language model to interpret the dependencies among terms inside a query. The experimental results showed that: Because of the nature of common user queries, such as short-length and no grammatical structure, traditional language modeling approaches (e.g., n-gram role language model) can not well capture the latent associates among terms inside a query. In contrast, the proposed simplified n-gram role language model integrates reasonable assumptions to model the dependency between the kernel-object and modifier, and achieves satisfactory performance when performing intent role annotation. Moreover, the high performances of the machine learning classifiers for role-explicit query identification demonstrate that: There are a set of effective features for differentiate the role-explicit queries from the role-implicit queries.

In Chapter 6, this thesis investigated how well the idea of intent role oriented query can work in reality. In particular, the strategy of subtopic mining via modifier graph clustering is proposed with respect to the hot topic of subtopic mining. The proposed framework for subtopic mining operates at a term granularity, which makes it easy to incorporate the term-level knowledge in query log. The reason is that: the click information for a whole query is relatively sparse, but the click information for a term is generally rich. Moreover, the term-level knowledge in query log is positive indicator for determining the subtopic relevance among terms. Towards this way, we quantified the edges of a modifier graph using the term-level knowledge in query log, the clustering of the modifier graph in turn uncovers the underlying subtopics among the modifiers. Based on the standard test collections released by NTCIR-10, the experimental results show that: the modifier graph based approaches outperform the two baseline methods that rely on the query-level knowledge in query log.

In summary, we can conclude that: The intent role oriented query parsing provides a flexible framework for query representation and understanding. Moreover, the proposed models and approaches discussed above pay the way for performing intent role oriented query parsing and adapting the intent role oriented query parsing into other applications.

7.2 Directions for Future Work

Throughout this thesis, we assume that there is one and only one kernel-object inside a role-explicit query. The more complicated cases of two or more kernel-objects has not been studied. In this way, a new series of query parsing schemes could be derived.

Analogous to the bag-of-word model, we viewed a role-explicit query as a bag of kernel-object and modifier, and each modifier is treated equally. The importance of the relative position of a modifier inside a query is also worthy to be investigated.

As discussed in Chapter 6, subtopic mining is a key step of search result diversification. The results of subtopic mining in this thesis show that the approaches based on modifier graph clustering exhibit stable performance even with the topics that has sparse query-level click information in a query log. However, whether the results of subtopic mining can improve the performance of search result diversification has not been investigated. Obviously, it is an interesting research direction to be further explored.

Overall, the future directions discussed above tie in with the intent role oriented query parsing, and make a great sense in applying the intent role oriented query parsing at a broader or larger scale.

Appendix A

Symbols Used in This Thesis

Symbol	Meaning
c	A class
ch	Character
d	Document (or web page)
ko	Kernel-object
mo	Modifier
q	Query
q^-	Role-implicit query
q^+	Role-explicit query
s	Session
s^-	Sin-session
s^+	Mul-session
t	Term
tp	Time stamp
u	User
w	Word
C	Cluster
CP	Click stamp
Q	Query set of a query log
G_{ko}	Modifier graph

Symbol	Meaning
U	User set of a query log
$\mathbb{V}$	Vocabulary
$\mathbb{D}$	Liguistic dictionary
Γ	Query log
$\ddot{t}$	Subtopic in the context of subtopic mining
$\ddot{T}$	Topic in the context of subtopic mining
$tStr$	Subtopic string
$\succ$	A composing unit of an element
$\in$	An element of a set

Table A.1: Symbols used throughout this thesis.

Bibliography

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the 2nd WSDM*, pages 5–14, 2009.

[2] N. O. Andrews and E. A. Fox. Recent developments in document clustering. In *Technical Report TR-07-35, Computer Science, Virginia Tech.*, 2007.

[3] Y. Azar, I. Gamzu, and X. Yin. Multiple intents re-ranking. In *Proceedings of the 41st STOC*, pages 669–678, 2009.

[4] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proceedings of the 13th KDD*, pages 76–85, 2007.

[5] P. Bailey, R. W. White, H. Liu, and G. Kumaran. Mining historic query trails to label long and rare search engine queries. *ACM Trans. Web*, 4(4):15:1–15:27, 2010.

[6] C. Barr, R. Jones, and M. Regelson. The linguistic structure of English web-search queries. In *Proceedings of the Conference on EMNLP*, pages 1021–1030, 2008.

[7] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: the Google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[8] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the 6th KDD*, pages 407–416, 2000.

[9] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st SIGIR*, pages 491–498, 2008.

[10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[11] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th CIKM*, pages 609–618, 2008.

[12] P. Boldi, F. Bonchi, C. Castillo, and S. Vigna. Query reformulation mining: models, patterns, and applications. *Journal of Information Retrieval*, 14(3):257–289, 2011.

[13] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. Recommendations for the long tail by term-query graph. In *Proceedings of the 20th WWW*, pages 15–16, 2011.

[14] F. Bonchi, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini. Efficient query recommendations in the long tail via center-piece subgraphs. In *Proceedings of the 35th SIGIR*, pages 345–354, 2012.

[15] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *In 11th Europ. Symp. Algorithms*, pages 568–579, 2003.

[16] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Journal of Computational Linguistics*, 18(4):467–479, 1992.

[17] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, pages 89–96, 2005.

[18] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th KDD*, pages 875–883, 2008.

[19] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *Proceedings of the 29th SIGIR*, pages 186–193, 2006.

[20] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th ICML*, pages 129–136, 2007.

[21] G. Capannini, F. M. Nardini, R. Perego, and F. Silvestri. Efficient diversification of web search results. *Proceedings of the VLDB Endowment*, 4(7):451–459, 2011.

[22] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th ACL*, pages 310–318, 1996.

[23] S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *Proceedings of the 14th WWW*, pages 2–11, 2005.

[24] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st SIGIR*, pages 659–666, 2008.

[25] C. L. Clarke, M. Kolla, and O. Vechtomova. An effectiveness measure for ambiguous and underspecified queries. In *Proceedings of the 2nd ICTIR*, pages 188–199, 2009.

[26] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), 2004.

[27] CNET. Another engine takes ads by the click. (http://news.com.com/Another engine takes ads by the click/2100-1033 _3-212736.html), 1996.

[28] ComScore. Comscore reports global search market growth of 46 percent in 2009. (http://comscore.com/Press_Events/Press_Releases/2010/1/Global_Search_Market_Grows_46_Percent_in_2009), 2009.

[29] E. David and K. Jon. *Networks, crowds, and markets: reasoning about a highly connected world*. Cambridge University Press, 2010.

[30] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[31] H. Deng, I. King, and M. R. Lyu. Entropy-biased models for query representation on the click graph. In *Proceedings of the 32nd SIGIR*, pages 339–346, 2009.

[32] L. Donetti and M. A. Muñoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004.

[33] Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th WWW*, pages 581–590, 2007.

[34] J. Du, Z. Zhang, J. Yan, Y. Cui, and Z. Chen. Using search session context for named entity recognition in query. In *Proceedings of the 33rd SIGIR*, pages 765–766, 2010.

[35] T. Finin, W. Murnane, A. Karandikar, N. Keller, J. Martineau, and M. Dredze. Annotating named entities in Twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on CSLDAMT*, pages 80–88, 2010.

[36] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1:385–408, 2004.

[37] S. Fortunato. Community detection in graphs. *CoRR*, abs/0906.0612, 2009.

[38] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie. Smoothing clickthrough data for web search ranking. In *Proceedings of the 32nd SIGIR*, pages 355–362, 2009.

[39] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th HLT*, pages 42–47, 2011.

[40] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*, pages 7821–7826, 2002.

[41] Y. Gotoh and S. Renals. Statistical language modelling. In *ELSNET Summer School*, pages 78–105, 2000.

[42] N. Gregoire, S. Evert, and S. N. Kim. Proceedings of the workshop on a broader perspective on multiword expressions. 2007.

[43] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proceedings of the 32nd SIGIR*, pages 267–274, 2009.

[44] M. Hagen, M. Potthast, B. Stein, and C. Bräutigam. Query segmentation revisited. In *Proceedings of the 20th WWW*, pages 97–106, 2011.

[45] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. In *Proceedings of the 15th KDD*, pages 10–18, 2009.

[46] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *Proceedings of the 35th SIGIR*, pages 305–314, 2012.

[47] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings of the 18th CIKM*, pages 77–86, 2009.

[48] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[49] B. J. Jansen and T. Mullen. Sponsored search: an overview of the concept, history, and technology. *International Journal of Electronic Business*, 6(2):114–131, 2008.

[50] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[51] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th KDD*, pages 133–142, 2002.

[52] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceedings of the 17th CIKM*, pages 699–708, 2008.

[53] G. D. F. Jr. The Viterbi algorithm: a personal history. *CoRR*, abs/cs/0504020, 2005.

[54] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, 1992.

[55] J.-W. Kuo, P.-J. Cheng, and H.-M. Wang. Learning to rank from Bayesian decision inference. In *Proceedings of the 18th CIKM*, pages 827–836, 2009.

[56] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proceedings of the 29th SFCS*, pages 422–431, 1988.

[57] G. Lembersky, N. Ordan, and S. Wintner. Language models for machine translation: original vs. translated texts. *Proceedings of the IEEE*, 38(4):799–825, 2012.

[58] X. Li. Understanding the semantic structure of noun phrase queries. In *Proceedings of the 48th ACL*, pages 1337–1345, 2010.

[59] Y. Li, B.-J. P. Hsu, C. Zhai, and K. Wang. Unsupervised query segmentation using clickthrough for information retrieval. In *Proceedings of the 34th SIGIR*, pages 285–294, 2011.

[60] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[61] M. Manshadi and X. Li. Semantic tagging of web search queries. In *Proceedings of the 47th ACL*, pages 861–869, 2009.

[62] D. W. Matula and F. Shahrokhi. Sparsest cuts and bottlenecks in graphs. *Discrete Applied Mathematics*, 27(1-2):113 - 123, 1990.

[63] N. Mishra, R. Saha Roy, N. Ganguly, S. Laxman, and M. Choudhury. Unsupervised query segmentation using only query logs. In *Proceedings of the 20th WWW*, pages 91–92, 2011.

[64] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[65] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(2), 2004.

[66] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8(1):1 – 38, 1994.

[67] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.

[68] M. Paşca. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the 16th CIKM*, pages 683–690, 2007.

[69] S. S. Piao, P. Rayson, D. Archer, and T. McEnery. Comparing and combining a semantic tagger and a statistical tool for MWE extraction. *Computer Speech and Language*, 19(4):378–397, 2005.

[70] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st SIGIR*, pages 275–281, 1998.

[71] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings of the 23rd SIGIR*, pages 184–191, 2000.

[72] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *Proceedings of the 29th SIGIR*, pages 691–692, 2006.

[73] F. Radlinski, M. Szummer, and N. Craswell. Inferring query intent from reformulations and clicks. In *Proceedings of the 19th WWW*, pages 1171–1172, 2010.

[74] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *Proceedings of the 19th WWW*, pages 781–790, 2010.

[75] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88:1270–1278, 2000.

[76] R. K. Roul and S. K. Sahay. An effective web document clustering for information retrieval. *CoRR*, abs/1211.1107, 2012.

[77] E. Sadikov, J. Madhavan, L. Wang, and A. Halevy. Clustering query refinements by user intent. In *Proceedings of the 19th WWW*, pages 841–850, 2010.

[78] R. Saha Roy, N. Ganguly, M. Choudhury, and S. Laxman. An IR-based evaluation framework for web search query segmentation. In *Proceedings of the 35th SIGIR*, pages 881–890, 2012.

[79] T. Sakai, Z. Dou, T. Yamamoto, Y. Liu, M. Zhang, and R. Song. Overview of the NTCIR-10 INTENT-2 task. In *Proceedings of NTCIR-10 Workshop Meeting*, pages 94–123, 2013.

[80] T. Sakai and R. Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of the 34th SIGIR*, pages 1043–1052, 2011.

[81] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Journal of Information Processing and Management*, 24(5):513–523, 1972.

[82] R. L. Santos, C. Macdonald, and I. Ounis. On the role of novelty for search result diversification. *Journal of Information Retrieval*, 15(5):478–502, 2012.

[83] S. E. Schaeffer. Stochastic local clustering for massive graphs. In *Proceedings of the 9th PAKDD*, pages 354–360, 2005.

[84] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[85] S. Sekine. Extended named entity ontology with attribute information. In *Proceedings of the 6th LREC*, 2008.

[86] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *SIGIR Forum*, pages 6–12, 1999.

[87] F. Silvestri. Mining query logs: turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, 4(1-2):1–174, 2010.

[88] F. Song and W. B. Croft. A general language model for information retrieval. In *Proceedings of the 8th CIKM*, pages 316–321, 1999.

[89] R. Song, M. Zhang, T. Sakai, M. P. Kato, Y. Liu, M. Sugimoto, Q. Wang, and N. Orii. Overview of the NTCIR-9 INTENT task. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 82–105, 2011.

[90] Y. Song and L.-w. He. Optimal rare query suggestion with implicit user feedback. In *Proceedings of the 19th WWW*, pages 901–910, 2010.

[91] Y. Song, D. Zhou, and L.-w. He. Query suggestion by constructing term-transition graphs. In *Proceedings of the 5th WSDM*, pages 353–362, 2012.

[92] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[93] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *Proceedings of the 17th WWW*, pages 347–356, 2008.

[94] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th KDD*, pages 718–723, 2006.

[95] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL*, pages 173–180, 2003.

[96] X. Wang, D. Chakrabarti, and K. Punera. Mining broad latent query aspects from search sessions. In *Proceedings of the 15th KDD*, pages 867–876, 2009.

[97] M. J. Welch and J. Cho. Automatically identifying localizable queries. In *Proceedings of the 31st SIGIR*, pages 507–514, 2008.

[98] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th WWW*, pages 162–168, 2001.

[99] Y. Xie and D. O. Hallaron. Locality in search engine queries and its implications for caching. In *Proceedings of the 21st INFOCOM*, 2002.

[100] G. Xu, S.-H. Yang, and H. Li. Named entity mining from click-through data using weakly supervised latent dirichlet allocation. In *Proceedings of the 15th KDD*, pages 1365–1374, 2009.

[101] J.-M. Yang, R. Cai, F. Jing, S. Wang, L. Zhang, and W.-Y. Ma. Search-based query suggestion. In *Proceedings of the 17th CIKM*, pages 1439–1440, 2008.

[102] X. Yi, H. Raghavan, and C. Leggetter. Discovering users' specific geo intention in web search. In *Proceedings of the 18th WWW*, pages 481–490, 2009.

[103] X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th WWW*, pages 1001–1010, 2010.

[104] H. Yu and F. Ren. Automatic role-explicit query extraction: A divide-and-conquer system leveraging on users' reformulating behaviors. *IEEJ Transactions on Electrical and Electronic Engineering*, 9(1), 2014.

[105] X. Yu and H. Shi. Query segmentation using conditional random fields. In *Proceedings of the 1st International Workshop on Keyword Search on Structured Data*, pages 21–26, 2009.

[106] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th SIGIR*, pages 10–17, 2003.

[107] C. Zhang, N. Sun, X. Hu, T. Huang, and T.-S. Chua. Query segmentation based on eigenspace similarity. In *Proceedings of the ACL-IJCNLP 2009 Conference*, pages 185–188, 2009.

[108] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *Proceedings of the 15th WWW*, pages 1039–1040, 2006.